Министерство образования и науки Российской Федерации
ФГБОУ ВО «Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий
Кафедра высшей математики

Использование системы Unity3D для моделирования решеток подалгебры конечномерных алгебр

Выпускная квалификационная работа

Квалификационная работа
допущена к защите
Зав. кафедрой

_____ _____
дата подпись

Исполнитель:
Смирнов Николай Алексеевич,
обучающийся БП-51Z
группы

_____
подпись

Руководитель ОПОП:

_____
подпись

Научный руководитель:
Коробков С.С.,
к.ф.-м.н., доцент

_____
подпись

Екатеринбург 2016

Оглавление

Введение

В выпускной квалификационной работе рассматривается алгебра $A = M_3(GF(2))$ квадратных матриц порядка 3 над полем $F = GF(2)$ из двух элементов $\{0, 1\}$. Основным объектом исследования являются подалгебры алгебры A и их решетки подалгебр. Плоское изображение диаграмм решеток становится мало информативным при большем количестве вершин и ребер соответствующих графов. Для повышения эффективности использования диаграмм решеток необходимо уменьшить количество самопересечений ребер диаграмм. Этого можно достичь, переходя к трехмерным изображениям диаграмм. Современные компьютерные пакеты, 3Ds Max, Blender3D и Unity3D и др. позволяют строить трехмерные виртуальные диаграммы. Перед визуализацией сложного математического объекта требуется создать сам объект с помощью специального математического пакета (например, с помощью системы компьютерной алгебры GAP). А далее требуется перенести полученные данные в пакеты для трехмерного моделирования. При этом возникает вопрос о совместимости математического пакета и пакета визуализации. Пакет Unity3D позволяет обработать математические данные (упорядоченные числовые последовательности). Последние обстоятельства определяет выбор пакета для визуализации.

Цель работы: разработать алгоритмы и программы для исследования решеток подалгебр алгебр малых размерностей и разработать алгоритмы и программы для визуализации полученных решеток.

Для достижения поставленных целей необходимо решить следующие задачи:

1) Освоить основные пакеты прикладных программ GAP и Unity3D;
2) Разработать алгоритмы и программы вычислений;
3) Разработать интерфейс для представления полученной информации;

Работа состоит из введения, трех глав, списка литературы и приложений.

Первая глава носит теоретический характер. В ней содержится определение алгебры над полем, приводятся примеры алгебр, а также понятие подалгебры

и признак подалгебры, приведено определение решетки, указаны основные свойства решеток. Здесь же приводится общая характеристика системы GAP и основные ее команды, также общая характеристика редактора Unity3D.

Вторая глава содержит практическую часть, где на примере подалгебры алгебры $M_3(GF(2))$ порожденной элементами (1, 193, 5) находится ее решетка и тип.

В третьей главе приведены алгоритмы создания программы для построения в пакете Unity3D трехмерной модели решетки подалгебры алгебры $M_3(GF(2))$ и описание работы созданной программы.

ГЛАВА I. **Теоретические основы**

1.1. Понятие алгебры над полем, примеры алгебр

Определение 2 [8]. Алгеброй над полем P называется множество A, на котором определены две бинарные операции + и ·, а также операция умножения элементов из P на элементы из A (то есть отображение $P \times A \to A$), удовлетворяющее следующим условиям:

1) $(A, +, \cdot)$ – кольцо;
2) $(A, +)$ – векторное пространство над полем P;
3) $\forall \alpha \in P \; \forall a, b \in A \; (\alpha a)b = \alpha(ab) = a(\alpha b)$.

Пример 1.

Пусть $M_n(F) = \{(\alpha_{ij}) \mid \alpha_{ij} \in F\}$ –множество всех квадратных матриц с коэффициентами из поля F. Ясно, что $M_n(F)$ – кольцо относительно операций сложения и умножения квадратных матриц. Если определить умножение элементов из F на элементы из $M_n(F)$ следующим образом:

$$\forall \beta \; \forall (\alpha_{ij}) \in M_n(F) \; \beta(\alpha_{ij}) = (\beta\alpha_{ij}),$$

то относительно такого умножения и сложения матриц множество $M_n(F)$ становится векторным пространством над полем F.

Пусть $a = (\alpha_{ij})$, $b = (\beta_{ij})$, $\alpha \in F$. Тогда $(\alpha a)b = (\alpha\alpha_{ij})(\beta_{ij}) = (\sum \alpha\alpha_{ij}\beta_{ij}) = (\sum \alpha_{ij}(\alpha\beta_{ij})) = a(\alpha b) = (\alpha(\sum \alpha_{ij}\beta_{ij})) = \alpha(ab)$.

Следовательно, $M_n(F)$ – алгебра над полем F. Эта алгебра называется *алгеброй матриц над полем F*.

Пример 2.

Пусть V – n-мерное пространство над полем F и $\Phi_n(F)$ - множество всех линейных преобразований пространства V. Известно, что - алгебра над полем F относительно следующих операций:

1) $\forall \varphi, \psi \Phi_n(F) \forall v \in V (\varphi + \psi)(v) = \varphi(v) + \psi(v)$;
2) $\forall \varphi, \psi \Phi_n(F) \forall v \in V (\varphi \circ \psi)(v) = \psi(\varphi(v))$;
3) $\forall \varphi, \psi \Phi_n(F) \forall \alpha \in F (\alpha\varphi(a) = \alpha(\varphi(a))$;

Известно также, что алгебра $\Phi_n(F)$ изоморфна алгебре $\mathrm{M}_n(F)$.

1.2. Алгебра матриц над полем

Определение 3 [8]. Пусть A — алгебра над полем P. Назовем алгебру A *конечномерной*, если A, как векторное пространство над полем P, *конечномерно*. При этом размерность векторного пространства A над P будем называть размерностью или рангом алгебры A.

Пример 3.

Пусть $A = C$, $P = R$. Тогда числа 1, i образуют базис C над R и потому $\dim C = 2$.

Пример 4.

Базис алгебры $M_n(F)$ образуют матричные единицы $E_{ij} = (e_{ij})$, где $e_{ij} = 0$, если $i \neq j$ и $e_{ij} = 1$, если $i = j$. Следовательно, $\dim M_n(F) = n^2$.

1.3. Понятие подалгебры, признак подалгебры

Определение 4 [8]. Подмножество S алгебры A над полем P назовем *подалгеброй алгебры A*, если относительно операций, определенных в A, S само является алгеброй над полем P.

Признак подалгебры: Непустое подмножество S алгебры A над полем P тогда и только тогда является подалгеброй в A, когда выполнены следующие условия:

1) $\forall a,b \in S\ a + b \in S$;
2) $\forall a,b \in S\ a \cdot b \in S$;
3) $\forall \alpha \in P\ \forall a \in S\ \alpha a \in S$.

Доказательство. Пусть S – подалгебра алгебры A. Тогда очевидно, что условия 1)- 3) выполнены. Обратно: пусть выполнены условия 1) - 3). Тогда из выполнимости условий 1) и 2) следует, что S – подкольцо кольца A, а из выполнимости условий 1) и 3) следует, что S—векторное подпространство пространства A. Условие

3) определения 3.1.выполняется в S, так как оно выполняется в A. Таким образом, S – подалгебра алгебры A.

1.4. Понятие решетки, основные свойства решеток

Определение 5 [6]. Верхней границей подмножества S частично упорядоченного множества (ч.у. множества) $(P, \leq)$ называется элемент $a \in P$, удовлетворяющий условию $(\forall s \in S\ s \leq a)$.

Определение 6 [6]. Верхней гранью (или супремумом) подмножества S ч. у. множества $(P, \leq)$ называется наименьший элемент в множестве верхних границ подмножества S.

Из определения 5 следует, что верхняя грань подмножества определяется однозначно (в отличие от верхней границы). Верхнюю грань подмножества S в подмножестве $T \subseteq P$ обозначают выражением $\sup_T S$, при этом индекс T, как правило, опускают, если $T = P$.

Определение 7 [6]. Двойственными понятиями верхней границы и верхней грани являются понятия *нижней границы* и *нижней грани*. Нижнюю грань называют также *инфимумом*. Нижнюю грань подмножества S в подмножестве $T \subseteq P$ обозначают символом $\inf_T S$ или символом $\inf S$ в случае, когда $T = P$. Двойственным образом устанавливается, что $\inf Ø$ существует тогда и только тогда, когда P содержит наименьший элемент l и что $\inf Ø = 1$.

Определение 8 [6]. Решеткой (или структурой) называется ч. у. множество, в котором каждое двухэлементное подмножество имеет нижнюю и верхнюю грани.

Определение 9 [6]. Полной решеткой называется ч. у. множество, в котором каждое подмножество имеет нижнюю и верхнюю грани.

Из этих определений следует, что любая полная решетка является решеткой. Но не наоборот.

Определение 9 [6]. Решеткой (или структурой) называется непустое множество L с определенными на нем двумя бинарными операциями ∨ и ∧, удовлетворяющими следующим условиям:

1. $\forall\, a \in L\ a \wedge a = a,\ a \vee a = a$ (идемпотентность);

2. $\forall\, a, b \in L\ a \wedge b = b \wedge a,\ a \vee b = b \vee a$ (коммутативность);

3. $\forall\, a, b, c \in L \quad a \vee (b \vee c) = (a \vee b) \vee c, \quad a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (ассоциативность);

4. $\forall\, a, b \in L\ a \vee (a \wedge b) = a \wedge (a \vee b) = a$ (поглощения).

Свойства решеток.

Лемма 1 [6]. Во всякой решетке $(L, \wedge, \vee)$ операции объединения и пересечения удовлетворяют условию изотонности: если $a \leq b$, то $a \wedge c \leq c \wedge b$ и $a \vee c \leq c \vee b$.

Лемма 2 [6]. Во всякой решетке $(L, \wedge, \vee)$ выполняются следующие неравенства дистрибутивности:

$(a \wedge b) \vee (a \wedge c) \leq a \wedge (b \vee c)$;

$a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$.

Лемма 3. Во всякой решетке $(L, \wedge, \vee)$выполняется неравенство модулярности: если $a \leq c$, то $a \vee (b \wedge c) \leq (a \vee b) \wedge c$.

1.5. Диаграммы решеток

Назовем элементы a и b ч.у. множества $(P, \leq)$ *сравнимыми*, если $a \leq b$ или $b \leq a$. Будем говорить, что элемент b покрывает элемент a, если выполнены следующие условия: 1) $a < b$; 2) $\forall c \in P\ ((a \leq c) \wedge (c \leq b) \rightarrow (c = a) \vee (c = b))$. Если b покрывает a, то будем записывать это кратко так: $a \prec b$.

В ряде случаев ч.у. множество может быть наглядно изображено в виде диаграммы на плоскости. Для того чтобы изобразить ч.у. множество $(P, \leq)$ в виде диаграммы, примем следующие соглашения:

1. Различные элементы множества P изображаются различными точками плоскости;
2. если $a, b \in P$ и b покрывает a, то точки, изображающие эти элементы, соединяются отрезком, причем точка, соответствующая b, располагается выше точки, соответствующей a.

Понятно, что диаграмма может быть построена полностью лишь в том случае, когда ч.у. множество P конечно. Однако при этом она может быть достаточно сложной и потому бесполезной. Очевидно также и то, что при построении диаграммы ее отрезки могут пересекаться в точках, не изображающих элементы множества P. Диаграмма, содержащая минимальное число таких пересечений, называется *оптимальной*, а не содержащая их совсем – *плоской*.

Пример 5. *$P = (M, \leq)$, где $M = \{1, 2, 5, 7\}$* (рис.1);

Пример 6. *$S = (M, |)$, где $M = \{2, 3, 4, 6, 12\}$* (рис.2).

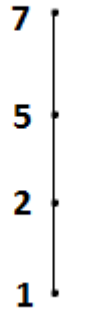

Рис.1

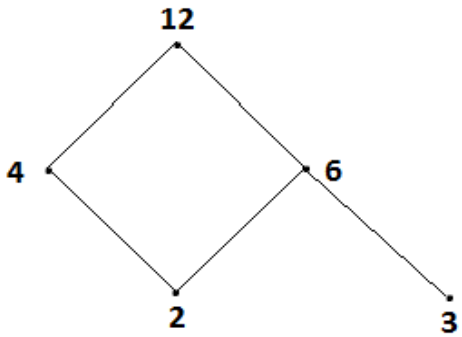

Рис.2

1.6. Алгебраические элементы колец

Теорема 1 [9]. Пусть $L(A)$ - алгебра над полем P. Тогда $(L(A), \wedge, \vee)$ - решетка.

Доказательство. Проверим выполнимость четырех аксиом решетки для $(L(A), \wedge, \vee)$.

1) Идемпотентность. $\forall B \in L(A)\ B \wedge B = B \cap B = B;\ B \vee B = \ = \ = B$

2) Коммутативность. $\forall B, C \in L(A)\ B \wedge C = B \cap C = C \cap B = C \wedge B;\ B \vee C = \ = \ = C \vee B.$

3) Ассоциативность. $\forall B, C, D \in L(A)\ B \wedge (C \wedge D) = B \cap (C \cap D) = (B \cap C) \cap D = (B \wedge C) \wedge D$ и $B \vee (C \vee D) = B \vee \ = \ = \ = \ = \ = \ \vee D = (B \vee C) \vee D.$

4) Поглощение. $\forall B, C \in L(A)\ B \wedge (B \vee C) = B \cap \ = B$ и $B \vee (B \wedge C) = B \vee (B \cap C) = B.$

1.7. Пирсовские разложения колец

Пусть K – коммутативное кольцо, e – ненулевой идемпотентный элемент.

Определим два множества:

1) $eK = \{ex | x \in K\} \neq \emptyset$

2) $(1-e)K = \{x - ex | x \in K\} \neq \emptyset$

Докажем, что eK и $(1-e)K$ – подкольца в K.

Признак подкольца:

1. $\forall a, b \in S\ \big((a-b) \in S\big)$

2. $\forall a, b \in S\ (ab \in S)$

Пусть $S = eK, a = ex_1, b = ex_2.$

1) $a - b = e(x_1 - x_2) \in eK$

2) $ab = ex_1 ex_2 = e^2 x_1 x_2 = e(ex_1 x_2) \in eK$

Пусть $S = (1-e)K, a = (1-e)x_1, b = (1-e)x_2.$

1) $a - b = x_1 - x_2 - e(x_1 - x_2) = (1-e)(x_1 - x_2) \in (1-e)K$

2) $ab = (1-e)x_1(1-e)x_2 = (1-e)((1-e)x_1 x_2) \in (1-e)K$

$eK + (1-e)K = K$ – докажем это:

Пусть $x \in K$. Тогда $x = ex + (1-e)x = ex + x - ex = x.$

Значит $K \subseteq eK + (1-e)K$. Так как $eK, (1-e)K \subseteq K$, то $eK + (1-e)K = \{ex + (1-e)y = ex + y - ey | x, y \in K\} \subseteq K.$

Убедимся в том, что $eK \cap (1-e)K = \{0\}$.

Пусть $a \in eK \cap (1-e)K$. Тогда $\exists\ x, y \in K$такие, что

$$a = ex = (1-e)y$$

$$ea = e(ex) = e(1-e)y = e(y-ey) = ey - ey = 0 \Rightarrow a = 0$$

Обозначим $eK + (1-e)K = eK \oplus (1-e)K$ – прямая сумма двух подколец.

Таким образом: $K = eK \oplus (1-e)K$ – пирсовское разложение кольца K по идемпотенту e. Легко видеть, что $\forall a \in eK\ ea,$ то есть e – единичный элемент в подкольце eK. Аналогично $\forall c \in (1-e)Kec = 0$. Значит, если $x \in eK$, а $y \in (1-e)K$, то $e(x+y) = x \Rightarrow xy = 0$.

Пусть K – некоммутативное кольцо, e – идемпотентный элемент. Тогда, если e – не единица, от имеет место двустороннее пирсовское разложение:

$$K = eKe \oplus eK(1-e) \oplus (1-e)Ke \oplus (1-e)K(1-e),$$

$$eKe \in a, eK(1-e) \in b, (1-e)Ke \in c, (1-e)K(1-e)d,$$

$$ba = 0, ab \in eK(1-e),$$

$$ac = 0, ca \in (1-e)Ke,$$

$$ad = da = 0,$$

$$b^2 = 0, c^2 = 0.$$

S – трехмерные. $S =< a_1, a_2, a_3 >, a_{1,}a_2, a_3 -$ базис. $\forall a \in S\ \exists \lambda_1, \lambda_2, \lambda_3 \in \{0,1\}\ a = \lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3. S =< e_1, e_2, r >, e_i^2 = e_i, r^2 = 0$

$$e_1 r = r, re_1 = 0, e_2 r = 0, re_2 = r, e_1 e_2 = e_2 e_1 = 0.$$

1.8. Общая характеристика пакета GAP

Система компьютерной алгебры GAP, название которой расшифровывается как "Groups, Algorithms and Programming", была задумана около 16 лет назад как инструмент комбинаторной теории групп – раздела алгебры, изучающего группы, заданные порождающими элементами и определяющими соотношениями. Однако с выходом каждой новой версии программы сфера ее применения охватывала все новые и новые разделы алгебры, и сейчас это довольно масштабная по своему охвату система. GAP наиболее привлекателен для исследований в области абстрактной алгебры.

Прежде всего, это возможность производить вычисления с гигантскими целыми и рациональными числами, допустимые значения которых ограничены только объемом доступной памяти. Так же система работает с конечными полями, многочленами от многих переменных, рациональными функциями, векторами и матрицами.

Группы могут быть заданы в различной форме, например, как группы подстановок, матричные группы, группы, заданные порождающими элементами и определяющими соотношениями. Функции для работы с группами включают определение порядка группы, вычисление классов сопряженных элементов. Для ряда конечных групп доступно определение их типа изоморфизма. В системе могут быть определены векторные пространства над всеми доступными полями.

Существует графический интерфейс XGAP, который позволяет, например, графически изобразить решетку подгрупп группы.

Основные особенности GAP:

- язык программирования, внешне напоминающий Паскаль;
- стандартные типы основных алгебраических объектов: групп (подстановок, абстрактных, матричных), колец, полей;
- удобные типы переменных, в т.ч. оперативно изменяемые списки и записи;
- более 4 тысяч библиотечных функций;
- обширная библиотека данных, включая практически все группы, порядок которых не превосходит 1000;
- прикладные программы, поставляемые вместе с **GAP**, охватывают такие разделы алгебры, как комбинаторная теория групп, конечные простые группы, теория представлений групп, теория графов, в т.ч. их группы автоморфизмов, теория кодирования, кристаллографические группы, группы Галуа и многое другое;
- подробное и удобное описание (около 1600 стр.) в формате «гипертекст»;

- бесплатное получение по сети Internet вместе с исходными текстами, являющимися незаменимым наглядным пособием для освоения GAP;
- работа в операционных системах DOS, Windows, Unix, Linux, MacOS;
- работа с процессором типа 386 и выше с ОЗУ от 8 Mb;
- занимаемое место на диске - от 10 до 100 Mb в зависимости от объема инсталляции;

способность работать с ОЗУ до 128 Mb и файлом подкачки до 128 Mb.

1.9. Общие команды пакета

Приведем список специфических команд системы GAP, используемых в программах данной работы:

Таблица 1

Список специфических команд системы GAP

MatAlgebra(GF(n),m)	# Построение алгебры матриц порядка *m* над полем, состоящим из *n* элементов
Elements(A)	# Элементы множества *A*
Dimension(A)	# Размерность алгебры *A*
Subalgebra(A,[m])	# Создание подалгебры алгебры *A*, порожденной элементом *m*
Работа со списками и множествами: N:=[]	# Создание пустого множества *N*
Size(N)	# Количество элементов множества *N*
AddSet(N,m)	# Присоединение элемента *m* к множеству *N*
Position(N,m)	# Порядок элемента *m* в списке *N*
IsSubsetSet(N,M)	# Проверяет, содержится ли каждый элемент множества *M* во множестве *N*
IntersectSet(N,M)	# Пересекает множество *N* с множеством *M*
UniteSet(N,M)	# Объединение множества *N* с множеством *M*
SubtractSet(N,M)	# Вычитает множество *M* от множества *N*, т.е. удаляет из множества *M* все элементы множества *N*
Условный оператор: if Q1 then P1; fi;	# Если *Q1*- истина, то выполняется команда *P1*

if Q1 then P1; elif Q2 then A2; fi;	# Если *Q1*- истина, то выполняется команда *P1*, а если *Q2* - истина, то выполняется команда *P2*
Работа с циклами: for a in N do P; od;	# Для всех элементов множества *N* выполняется команда *P*
for i in [1..m] do P; od;	# Выполнение команды *P m* раз
Работа с данными: PrintTo(“***.dan’’, N)	# Записывает данные в файл
Read(“***.dan’’)	# Читает файл
quit;	# Выход из программы

1.10. Команды для вычислений в алгебрах

Пусть F – поле и A – алгебра над F (кратко: A – F -алгебра). Все алгебры в GAPe ассоциативны, то есть операция умножения в них ассоциативна. Любая алгебра все- гда содержит нулевой элемент, который может быть получен, вычитанием произволь- ного элемента из самого себя. Элементы поля F не рассматриваются как элементы A. Практическим обоснованием (очевидно и математическим тоже) для этого служит то, что даже если единичная матрица содержится в матричной алгебре A, все равно не возможно записать 1 + a для суммы единичной матрицы и элемента a алгебры A, так как независимо от алгебры A в GAPe уже определено это значение как прибавление 1 ко всем позициям матрицы a. Вместо этого необходимо писать One(A) + a или aˆ0 + a.

Алгебры и унитальные алгебры

Не все алгебры содержат (левый и правый) мультипликативный нейтральный единичный элемент, но если алгебра содержит такой единичный элемент, то он единственен. Если алгебра A содержит мультипликативный нейтральный элемент, то в общем случае он не может быть получен из произвольного элемента a алгебры A как частное a/a или как a0, так как эти операции могут быть не определены для алгебры A. Более точно, может быть возможно инвертировать a или возвести его в нулевую степень, но A может быть не замкнуто относительно этих операций. Например, если a - квадратная матрица в GAPe, тогда мы можем считать, что a0 является единичной матрицей того же самого размера и над тем же

самым полем, что и a. С другой стороны, алгебра может иметь мультипликативный нейтральный элемент, который не равен нулевой степени элементов. В многих случаях, однако, нулевая степень элементов алгебры правильно определена как элемент алгебры. Это справедливо, например, для для всех тех матричных алгебр, чьи порождающие элементы являются порождающими элементами конечной группы.

Для практических целей полезно различить общие алгебры и унитальные агебры.

Унитальная алгебра в GAPe – алгебра U , которая содержит нулевую степень элементов, и в которой выполнимы все действия с этой степенью. Не унитальная алгебра A может не содержать нулевые степени элементов или, наоборот содержать их, но тогда в ней невыполнимо какое-нибудь действие с этой степенью. Итак, возможно рассматривать A как унитальную алгебру, используя AsUnitalAlgebra (A), и, конечно, всегда возможно рассматривать унитальную алгебру как алгебру, используя AsAlgebra (U). Алгебра A может иметь унитальные подалгебры, и, конечно, алгебра U может иметь подалгебры, которые не являются унитальными. Образы унитальных алгебр при гjоморфизмах являются или унитальными, или тривиальными, так как образ единицы при гомоморфизме есть единица. Следующий пример показывает главные различия между алгебрами и унитальными алгебрами.

```
gap> a:= [ [ 1, 0 ], [ 0, 0 ] ];;
gap> alg1:= Algebra( Rationals, [ a ] );
Algebra( Rationals, [ [ [ 1, 0 ], [ 0, 0 ] ] ] )
gap> id:= a^0;
[ [ 1, 0 ], [ 0, 1 ] ]
gap> id in alg1;
false
```

gap> alg2:= UnitalAlgebra(Rationals, [a]); UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]]) gap> id in alg2;

true

gap> alg3:= AsAlgebra(alg2);

Algebra(Rationals, [[[1, 0], [0, 0]], [[1, 0], [0, 1]]

])

gap> alg3 = alg2;

true

gap> AsUnitalAlgebra(alg1); Error, <D> is not unital

Мы видим, что, если мы хотим, чтобы единичная матрица содержалась в алгебре, о которой неизвестно является ли она унитальной, нам необходимо прибавить ее к по- рождающим элементам.

Родительские алгебры и подалгебры

GAP различает алгебры и подалгебры алгебр. Каждая подалгебра принадлежит уникальной основной алгебре, которую называют родителем подалгебры. Родительская алгебра – собственный родитель. Родительские алгебры конструируются при помощи операторов Algebra и UnitalAlgebra, подалгебры конструируются при помощи операторов Subalgebra и UnitalSubalgebra. Родитель первого аргумента оператора Subalgebra будет родителем созданной подалгебры. Алгебраические действия, совершаемые более, чем с од- ной алгеброй, предполагают, что аргументы имеют общего родителя. Возьмем, напри- мер, Centralizer. В этом случае должно быть два аргумента: алгебра A и алгебра B, где A родительская алгебра и B – подалгебра этой родительской алгебры, или A и B – подалгебры общей родительской алгебры P . В этих случаях Centralizer выдает централизатор B в A, который представлен как подалгебра общей родительской алгебры алгебр A и B. Заметим, что подалгебра родительской алгебры не должна быть собственной подалгеброй. Исключением этому правилу является теоретико-множественная функция Intersection, которая

позволяет рассматривать пересечения алгебры с различными родительскими алгебрами. Всякий раз, когда имеется две подалгебры, которые имеют различные родительские алгебры, но имеют и общую супералгебру A, можно использовать AsSubalgebra или AsUnitalSubalgebra для того, чтобы создать новые подалгебры, которые имеют общую родительскую алгебру A.

Algebra

Algebra(U) – выдает родительскую алгебру A, которая изоморфна родительской алгебре или подал- гебре U .

Algebra(F, gens); Algebra(F,gens, zero) – выдает родительскую алгебру над полем F и порожденную элементами алгебры в списке gens. Нулевой элемент этой алгебры может быть введен как zero; это необходимо всякий раз, когда gens пусто.

```
gap> a:= [ [ 1 ] ];;
gap> alg:= Algebra( Rationals, [ a ] ); Algebra( Rationals, [ [ [ 1 ] ] ] )
gap> alg.name:= "alg";;
gap> sub:= Subalgebra( alg, [] ); Subalgebra( alg, [ ] )
gap> Algebra( sub );
```

Algebra(Rationals, [[[0]]]) gap> Algebra(Rationals, [], 0*a); Algebra(Rationals, [[[0]]])

Алгебры, получаемые с помощью Algebra, не унитальны. Для построения унитальных алгебр используйте UnitalAlgebra.

UnitalAlgebra

UnitalAlgebra (U) – выдает унитальную родительскую алгебру A, которая изоморфна родительской алгебре или подалгебре U . Если U – не унитальная, то проверяется, содержится ли нулевая степень элементов в U , и если не содержится, то выдается сообщение об ошибке.

UnitalAlgebra(F, gens) UnitalAlgebra(F, gens, zero) – выдает унитальную родительскую алгебру над полем F и порожденную алгебраическими элементами, указанными в списке gens. Нулевой элемент этой алгебры может быть введен как zero ; это необходимо всякий раз, когда gens пусто.

gap> alg1:= UnitalAlgebra(Rationals, [NullMat(2, 2)]);

UnitalAlgebra(Rationals, [[[0, 0], [0, 0]]])

gap> alg2:= UnitalAlgebra(Rationals, [], NullMat(2, 2)); UnitalAlgebra(Rationals, [[[0, 0], [0, 0]]])

gap> alg3:= Algebra(alg1);

Algebra(Rationals, [[[0, 0], [0, 0]], [[1, 0], [0, 1]]

])

gap> alg1 = alg3;

true

gap> AsUnitalAlgebra(alg3);

UnitalAlgebra(Rationals, [[[0, 0], [0, 0]], [[1, 0], [0,

1]]])

Алгебры, введенные при помощи UnitalAlgebra являются унитальными. Для построения неунитальных алгебр используйте Algebra.

IsAlgebra

IsAlgebra (obj) – выдает true, если obj, который может быть объектом произвольного типа, является родительской алгеброй или подалгеброй и false в противном случае. Эта функция сообщит об ошибке, если obj – неограниченная переменная.

gap> IsAlgebra(FreeAlgebra(GF(2), 0));

true

gap> IsAlgebra(1/2);

false

IsUnitalAlgebra

IsUnitalAlgebra (obj) – выдает true, если obj, который может быть объектом произвольного типа, является унитальной родительской алгеброй или унитальной подалгеброй, и false в противном случае. Эта функция сообщит об ошибке, если obj – неограниченная переменная.

```
gap> IsUnitalAlgebra( FreeAlgebra( GF(2), 0 ) );
true
gap> IsUnitalAlgebra( Algebra( Rationals, [ [ [ 1 ] ] ] ) );
false
```

Заметьте, что эта функция не проверяет является ли obj алгеброй, которая содержит нулевую степень элементов, но только проверяет является ли obj алгебра с флажком isUnitalAlgebra.

Subalgebra

Subalgebra(A, gens) – выдает подалгебру алгебры A, порожденную элементами в списке gens.

```
gap> a:= [ [ 1, 0 ], [ 0, 0 ] ];;
gap> b:= [ [ 0, 0 ], [ 0, 1 ] ] ;;
gap> alg:= Algebra( Rationals, [ a, b ] );;
gap> alg.name:= "alg";;
gap> s:= Subalgebra( alg, [ a ] );
Subalgebra( alg, [ [ [ 1, 0 ], [ 0, 0 ] ] ] )
gap> s = alg;
false
gap> s:= UnitalSubalgebra( alg, [ a ] ); UnitalSubalgebra( alg, [ [ [ 1, 0 ], [ 0, 0 ] ] ] ) gap> s = alg;
true
```

UnitalSubalgebra

UnitalSubalgebra (A, gens) – выдает унитальную подалгебру алгебры A, порожденную элементами в списке gens. Если неизвестно является ли A унитальной, то сначала проверяется содержит ли A нулевую степень элементов.

gap> a:= [[1, 0], [0, 0]];;

gap> b:= [[0, 0], [0, 1]] ;;

gap> alg:= Algebra(Rationals, [a, b]);;

gap> alg.name:= "alg";;

gap> s:= Subalgebra(alg, [a]);

Subalgebra(alg, [[[1, 0], [0, 0]]])

gap> s = alg;

false

gap> s:= UnitalSubalgebra(alg, [a]); UnitalSubalgebra(alg, [[[1, 0], [0, 0]]]) gap> s = alg;

true

IsSubalgebra

IsSubalgebra (A, U) – выдает true, если U – подалгебра A и false в противном случае.

Заметьте, что A и U должны иметь общую родительскую алгебру. Эта функция вы- дает true, тогда и только, когда множество элементов U – подмножество множества элементов A.

gap> a:= [[1, 0], [0, 0]];;

gap> b:= [[0, 0], [0, 1]] ;;

gap> alg:= Algebra(Rationals, [a, b]);;

gap> alg.name:= "alg";;

gap> IsSubalgebra(alg, alg);

true

gap> s:= UnitalSubalgebra(alg, [a]); UnitalSubalgebra(alg, [[[1, 0], [0, 0]]]) gap> IsSubalgebra(alg, s);

true

AsAlgebra

AsAlgebra (D) AsAlgebra (F, D)

Пусть D – область. AsAlgebra выдает алгебру A над полем F такую, что множество элементов D является также множеством элементов A, если это возможно. Если D – алгебра, аргумент F может быть опущен, в этом случае в качестве поля коэффициентов D принимается поле F.

Если D – список элементов алгебры, эти элементы должны порождать алгебру, в противном случае выдается сообщение об ошибке.

gap> a:= [[1, 0], [0, 0]] * Z(2);;

gap> AsAlgebra(GF(2), [a, 0*a]);

Algebra(GF(2), [[[Z(2)^0, 0*Z(2)], [0*Z(2), 0*Z(2)]]]) Заметьте, что эта функция выдает родительскую алгебру или подалгебру родительской алгебры в зависимости от D. Чтобы преобразовывать подалгебру в родительскую алгебру, нужно использовать Algebra или UnitalAlgebra.

AsUnitalAlgebra

AsUnitalAlgebra (D) AsUnitalAlgebra (F, D)

Пусть D – область. AsUnitalAlgebra выдает унитальную алгебру A над полем F та- кую, что множество элементов D является также множеством элементов A, если это возможно. Если D – алгебра, аргумент F может быть опущен, в этом случае в качестве поля коэффициентов D принимается поле F.

Если D – список элементов алгебры, эти элементы должны порождать унитальную алгебру, в противном случае выдается сообщение об ошибке.

gap> a:= [[1, 0], [0, 0]] * Z(2);;

gap> AsUnitalAlgebra(GF(2), [a, a^0, 0*a, a^0-a]); UnitalAlgebra(GF(2), [[[0*Z(2), 0*Z(2)], [0*Z(2), Z(2)^0]], [[Z(2)^0, 0*Z(2)], [0*Z(2), 0*Z(2)]]])

Заметьте, что эта функция выдает родительскую алгебру или подалгебру родительской алгебры в зависимости от D. Чтобы преобразовывать подалгебру в родительскую алгебру, нужно использовать Algebra или UnitalAlgebra.

AsSubalgebra

AsSubalgebra (A, U)

Пусть A – родительская алгебра и U – родительская алгебра или подалгебра воз- можно другой родительской алгебры, такой, что порождающие элементы U являются элементами A. AsSubalgebra выдает новую подалгебру S такую, что S имеет родительскую алгебру A и порожденную образующими элементами алгебры U.

```
gap> a:= [ [ 1, 0 ], [ 0, 0 ] ];;
gap> b:= [ [ 0, 0 ], [ 0, 1 ] ] ;;
gap> alg:= Algebra( Rationals, [ a, b ] );;
gap> alg.name:= "alg";;
gap> s:= Algebra( Rationals, [ a ] );
Algebra( Rationals, [ [ [ 1, 0 ], [ 0, 0 ] ] ] )
gap> AsSubalgebra( alg, s );
Subalgebra( alg, [ [ [ 1, 0 ], [ 0, 0 ] ] ] )
AsUnitalSubalgebra
AsUnitalSubalgebra (A, U)
```

Пусть A – родительская алгебра и U – родительская алгебра или подалгебра с воз- можно другой родительской алгеброй, такой, что порождающие элементы (сокращен- но: генераторы) U являются элементами A. AsSubalgebra выдает новую унитальную подалгебру S такую, что S имеет родительскую алгебру A и порожденную генератора- ми U . Если U или A не содержат нулевых степеней элементов, то выдается сообщение об ошибке.

```
gap> a:= [ [ 1, 0 ], [ 0, 0 ] ];;
gap> b:= [ [ 0, 0 ], [ 0, 1 ] ];;
gap> alg:= Algebra( Rationals, [ a, b ] );;
gap> alg.name:= "alg";;
gap> s:= UnitalAlgebra( Rationals, [ a ] ); UnitalAlgebra( Rationals, [ [ [ 1, 0 ],
[ 0, 0 ] ] ] ) gap> AsSubalgebra( alg, s );
Subalgebra( alg, [ [ [ 1, 0 ], [ 0, 0 ] ], [ [ 1, 0 ], [ 0, 1 ] ] ] )
gap> AsUnitalSubalgebra( alg, s );
```

UnitalSubalgebra(alg, [[[1, 0], [0, 0]]])

Нуль и единица для алгебр

Zero(A) выдает аддитивный нейтральный элемент алгебры A.

One(A) выдает (правый и левый) мультипликативный нейтральный элемент алгебры A, если он существует, и false в противном случае. Если A – унитальная алгебра, то этот элемент получен при возведении в нулевую степень произвольного элемента.

gap> a:= Algebra(Rationals, [[[1, 0], [0, 0]]]); Algebra(Rationals, [[[1, 0], [0, 0]]])

gap> Zero(a);

[[0, 0], [0, 0]]

gap> One(a);

[[1, 0], [0, 0]]

gap> a:= UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]]); UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]])

gap> Zero(a);

[[0, 0], [0, 0]]

gap> One(a);

[[1, 0], [0, 1]]

Теоретико-множественные функции для алгебр

Как уже упомянуто во введении главы, алгебры являются областями. Таким образом, все теоретико-множественные функции, например, Intersection и Size могут быть применены к алгебрам. Все теоретико-множественные функции, не упомянутые здесь, не трактуются специально для алгебр.

Elements(A) вычисляет элементы алгебры A с использованием алгоритма Dimino. Заданная по умолчанию для алгебр функция вычисляет базис линейного пространства в то же самое время.

Intersection(A, H) выдает пересечение A и H в виде множества элементов или как алгебраическую запись (запись алгебры).

IsSubset (A, H)

Если A и H – алгебры, то IsSubset проверяет являются ли генераторы H элементами A. Другой способ состоит в применении DomainOps.IsSubset.

Random(A) выдает произвольный элемент алгебры A. Это требует вычисления базиса линейного пространства.

Проверка свойств алгебр

С помощью GAPa могут быть проверены следующие свойства алгебр.

IsAbelian(A) выдается true если алгебра A абелева и false в противном случае. Алгебра A назы- вается абелевой, если и только, если для любых a, b ∈ A a ∗ b = b ∗ a.

IsCentral(A, U) выдается true если алгебра A централизует алгебру U и false в противном случае. Алгебра A централизует алгебру U , если и только, если для всякого a ∈ A и для всякого u ∈ U a ∗ u = u ∗ a. Заметьте, что U не обязана быть подалгеброй A, но они должны иметь общую родительскую алгебру.

IsFinite(A) выдается true если алгебра A конечна, и false в противном случае.

IsTrivial(A) выдается true если алгебра A состоит только из нулевого элемента, и false в против- ном случае. Если A – унитальная алгебра, то, конечно, она никогда не тривиальна.

Все критерии ожидают родительскую алгебру или подалгебру и выдают true, ес- ли алгебра имеет свойство и false в противном случае. Некоторые функции не могут выполниться, если данная алгебра имеет бесконечное множество элементов. В таких случаях может быть напечатано предупреждение.

gap> IsAbelian(FreeAlgebra(GF(2), 2));

false

gap> a:= UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]]); UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]])

gap> a.name:= "a";;

gap> s1:= Subalgebra(a, [One(a)]); Subalgebra(a, [[[1, 0], [0, 1]]]) gap> IsCentral(a, s1); IsFinite(s1); true

false

gap> s2:= Subalgebra(a, []); Subalgebra(a, [ ])

gap> IsFinite(s2); IsTrivial(s2);

true true

Функции линейного пространства для алгебр

Конечномерная F -алгебра A всегда есть конечно-мерное векторное пространство над F . Таким образом, в GAPe, алгебра – линейное пространство, и функции линейного пространства типа Base и Dimension применимы к алгебрам.

gap> a:= UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]]); UnitalAlgebra(Rationals, [[[1, 0], [0, 0]]])

gap> Dimension(a);

gap> Base(a);

[[[1, 0], [0, 1]], [[0, 0], [0, 1]]]

Структура линейного пространства используется также теоретико-множественными функциями.

Алгебраические функции для алгебр

Функции, описанные в этом разделе, вычисляют некоторые подалгебры данной алгебры, например, Centre вычисляет центр алгебры. Некоторые функции не могут завершиться, если данная алгебра имеет бесконечное множество элементов, в то время как другие функции могут сообщить об ошибке в таких случаях.

В GAPe каждая алгебра является или родительской алгеброй или подалгеброй единственной родительской алгебры. Если Вы вычисляете центр C алгебры U с родительской алгеброй A, то C – подалгебра U , но ее родительская алгебра есть A.

Centralizer(A, x)

Centralizer(A, U) выдают централизатор элемента x в A, где x должен быть элементом родительской алгебры A, соответственно централизатор алгебры U в A, где обе алгебры должны иметь общего родителя.

Централизатор элемента x в A определен как множество C элементов c из A, таких, что c и x коммутируют.

Централизатор алгебры U в A определен как множество C элементов c из A, таких, что c коммутирует с каждым элементом U .

```
gap> a:= MatAlgebra( GF(2), 2 );;
gap> a.name:= "a";;
gap> m:= [ [ 1, 1 ], [ 0, 1 ] ] * Z(2);;
gap> Centralizer( a, m );
UnitalSubalgebra( a, [ [ [ Z(2)^0, 0*Z(2) ], [ 0*Z(2), Z(2)^0 ] ], [ [ 0*Z(2), Z(2)^0
], [ 0*Z(2), 0*Z(2) ] ] ] )
```

Centre(A) выдает центр A (то есть централизатор A в A).

```
gap> c:= Centre( a );
UnitalSubalgebra( a, [ [ [ Z(2)^0, 0*Z(2) ], [ 0*Z(2), Z(2)^0 ] ]
] )
```

Closure(U , a) Closure(U , S)

Пусть U – алгебра с родительской алгеброй A и пусть a – элемент A. Тогда Closure выдает замыкание C алгебры U и элемента a как подалгебру алгебры A. Замыкание C алгебры U и элемента a – подалгебра, порожденная U и a.

Пусть U и S две алгебры с общей родительской алгеброй A. Тогда Closure выдает подалгебру A, порожденную U и S.

```
gap> Closure( c, m );
UnitalSubalgebra( a, [ [ [ Z(2)^0, 0*Z(2) ], [ 0*Z(2), Z(2)^0 ] ], [ [ Z(2)^0, Z(2)^0
], [ 0*Z(2), Z(2)^0 ] ] ] )
```

TrivialSubalgebra

TrivialSubalgebra(U)

Пусть U – алгебра с родительской алгеброй A. Тогда TrivialSubalgebra выдает тривиальную подалгебру T алгебры U , как подалгебру алгебры A.

gap> a:= MatAlgebra(GF(2), 2);;

gap> a.name:= "a";;

gap> TrivialSubalgebra(a); Subalgebra(a, [ ])

Элементы алгебры

Этот раздел описывает операции и функции, доступные для элементов алгебры.

Заметьте, что элементы алгебры могут существовать независимо от алгебры, например, вы можете записывать две матрицы и вычислять их сумму и произведение без когда-либо определения алгебры, которая содержит их.

Сравнения элементов алгебры

g = h выдает true, если элементы алгебры g и h равны и false в противном случае.

g <> h выдает true, если элементы алгебры g и h не равны и false в противном случае.

g < h

g <= h g >= h g > h

Операторы <, <=, >= и > выдают true, если элемент g – строго меньше, меньше или равен, больше или равен и строго больше, чем элемент h. Общего упорядочения всех элементов алгебры может не быть, но g и h должны лежать в одной родительской алгебре. Арифметические операции для элементов алгебры

a ∗ b

a + b a – b

Операторы * , + и - вычисляют произведение, сумму и разность двух элементов алгебры a и b. Операнды должны лежать в общей родительской алгебре, в противном случае выдается сообщение об ошибке.

a/c

выдает частное элемента a и ненулевого элемента c основного поля алгебры (поля частных алгебры?).

a ^i

выдает i-ую степень элемента a для положительного целого числа i. Если число i – нуль или отрицательно, то возможно результат не определен, или не содержится в алгебре, порожденной элементом a.

list + a a + list

list ∗ a a ∗ list

В этой форме операторы + и * выдают новый список, где каждая запись – сумма и соответственно произведение элемента a и соответствующего элемента списка. Конечно сложение и соответственно умножение должно быть определено между a и каждым элементом списка.

IsAlgebraElement

IsAlgebraElement (obj) – выдает true, если obj, который может быть объектом произвольного типа, является элементом алгебры, и false в противном случае. Функция сообщит об ошибке, если obj – недопустимая переменная.

```
gap> IsAlgebraElement( (1,2) );
false
gap> IsAlgebraElement( NullMat( 2, 2 ) );
true
gap> IsAlgebraElement( FreeAlgebra( Rationals, 1 ).1 );
true
```

MatAlgebra

MatAlgebra(F, n) – выдает полную матричную алгебру, состоящую из n × n-матриц над полем F.

```
gap> a:= MatAlgebra( GF(2), 2 );
UnitalAlgebra( GF(2), [ [ [ Z(2)^0, 0*Z(2) ], [ 0*Z(2), 0*Z(2) ] ], [ [ 0*Z(2),
Z(2)^0 ], [ Z(2)^0, 0*Z(2) ] ] ] )
```

gap> Size(a);

1.11. Редактор Unity3D

Unity3D - это инструмент для разработки двух- и трёхмерных приложений и игр, работающий под операционными системами Windows и OS X. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL. Ранее была экспериментальная поддержка реализации проектов в рамках модуля Adobe Flash Player, но позже команда разработчиков Unity приняла сложное решение по отказу от этого.

Приложения, созданные с помощью Unity, поддерживают DirectX и OpenGL. Активно движок используется как крупными разработчиками (Blizzard, EA, QuartSoft, Ubisoft), так и девелоперами Indie-игр (например, ремейк Mop. Утопия (Pathologic), Kerbal Space Program, Slender: The Eight Pages, Slender: The Arrival, Surgeon Simulator 2013, Baeklyse Apps: Save the Bubble и т. п.) в силу наличия бесплатной версии, удобного интерфейса и простоты работы с движком.

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок поддерживает три сценарных языка: C# (на данном языку будут написаны все скрипты для работоспособности нашего проекта), JavaScript (модификация), Boo (диалект Python). Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены (уровни) – отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые

объекты – объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковым названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform – он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

К объектам можно применять коллизии (в Unity т. н. коллайдеры – collider). Существует несколько типов коллайдеров:

1) Character controller – вид физической модели, созданный специально под использование его для игровых персонажей;
2) Box collider (физическая модель образует куб, в который попадает вся модель объекта);
3) Sphere collider (физическая модель образует сферу, в которую попадает вся модель объекта);
4) Capsule collider (физическая модель образует капсулу, в которую попадает модель объекта. В отличие от предыдущего типа размеры можно менять и по одной, и по трём осям сразу);
5) Mesh collider (физическая модель полностью повторяет реальную геометрию объекта);
6) Wheel collider (физическая модель колеса);

Terrain collider – тип физической модели, созданный специально для использования на объекте типа Terrain – земля, генерируемая редактором Unity с возможностями скульптинга и окрашивания местности.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя – будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели. Редактор Unity поддерживает написание и редактирование шейдеров. Редактор Unity имеет компонент для создания анимации, но также анимацию можно создать предварительно в 3D-редакторе и импортировать вместе с моделью, а затем разбить на файлы.

Помимо пустого игрового объекта и моделей, на сцену можно добавлять ещё такие объекты типа GameObject:

1) Система частиц;
2) Камера;
3) GUI текст;
4) GUI текстура;
5) 3D текст;
6) Точечный свет;
7) Направленный свет;
8) Освещение территории;
9) Источник света, имитирующий солнце;
10) Стандартные примитивы;
11) Деревья;
12) Terrain (земля).

Unity 3D поддерживает систему Level Of Detail (сокр. LOD), суть которой заключается в том, что на дальнем расстоянии от игрока высокодетализированные модели заменяются на менее детализированные, и наоборот, а также систему

Occlusion culling, суть которой в том, что у объектов, не попадающих в поле зрения камеры не визуализируется геометрия и коллизия, что снижает нагрузку на центральный процессор и позволяет оптимизировать проект. При компиляции проекта создается исполняемый (.exe) файл игры (для Windows), а в отдельной папке – данные игры (включая все игровые уровни и динамически подключаемые библиотеки).

Движок поддерживает множество популярных форматов, таких как:

1) .3ds, .max, .obj, .fbx, .dae, .ma, .mb для трёхмерных моделей;
2) .mp3, .wmv, .ogg для звуковых файлов;
3) .bmp, .gif, .png, .tga, .psd, .tif, .dds для изображений;
4) .mov, .ovg для видеофайлов.

Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат .unityassets и передавать другим разработчикам, или выкладывать в свободный доступ. Этот же формат используется во внутреннем магазине Unity Asset Store, в котором разработчики могут бесплатно и за деньги выкладывать в общий доступ различные элементы, нужные при создании игр. Чтобы использовать Unity Asset Store, необходимо иметь аккаунт разработчика Unity. Unity имеет все нужные компоненты для создания мультиплеера. Также можно использовать подходящий пользователю способ контроля версий. К примеру, Tortoise SVN или Source Gear.*подалгеброй*

ГЛАВА II. Типовая классификация подалгебр алгебры матриц M(GF(2),3)

2.1. Программа 1 Создадим массив порождающих элементов в пакете GAP

№	Текст программы	Комментарии
1	Al:=MatAlgebra(GF(2),3);	Построение алгебры матриц третьего порядка над полем GF(2)
2	El:=Elements(Al);	Создание массива элементов алгебры Al
3	A:=[];	Создаем массив
4	C:=[];	Создаем массив
5	for i in [1..512] do	Начало цикла
6	S:=Subalgebra(Al,[El[i]]);	Записывает подалгебру алгебры Al порожденную элементам El[i] в S
7	el:=Elements(S);	Создание массива элементов подалгебры S
8	a:=Size(A);	Проверяем размер массива A
9	AddSet(A,el);	Заполняем массив A элементами из массива el
10	b:=Size(A);	Проверяем размер массива A
11	if b > a then	Проверяем наличие новых элементов
12	Add(C,i);	Если есть новые, то добавляем элемент алгебры Al в массив C
13	fi;	Конец условия
14	od;	Конец цикла

15	PrintTo("Dann.dan","Gen:=",C,";");	Записываем полученные данные в файл

Результат выполнения:

Gen:=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41,
42, 43, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 61, 62, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102,
103, 104, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 118, 119,
120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134,
136, 137, 138, 139, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
155, 159, 160, 161, 162, 163, 164, 165, 167, 168, 169, 170, 171, 174, 175,
176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 190, 192, 193,
194, 196, 197, 198, 199, 200, 202, 203, 206, 207, 209, 210, 211, 212, 214,
215, 216, 217, 218, 220, 221, 222, 223, 224, 225, 228, 229, 232, 233, 234,
235, 236, 238, 239, 240, 241, 242, 244, 246, 247, 249, 250, 252, 253, 255,
257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271,
272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 284, 285, 286, 287,
288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 300, 301, 302, 303,
305, 306, 307, 308, 309, 310, 311, 313, 314, 316, 317, 320, 321, 322, 323,
324, 326, 328, 329, 330, 331, 332, 334, 337, 338, 339, 340, 342, 344, 345,
346, 347, 348, 350, 351, 353, 354, 361, 362, 363, 366, 367, 368, 369, 370,
372, 374, 375, 377, 382, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394,
396, 399, 401, 402, 403, 404, 405, 406, 407, 409, 410, 412, 428, 431, 433,
434, 435, 436, 438, 439, 440, 441, 442, 447, 449, 450, 451, 452, 456, 457,
458, 459, 463, 464, 465, 466, 467, 470, 472, 473, 476, 477, 485, 488, 491,
494, 495, 498, 503, 505, 506, 507, 509, 512];

Известно, что любая подалгебра в алгебре матриц M(GF(2),3) порождается не более, чем тремя элементами. Следующая программа позволяет найти тройки матриц, порождающих все подалгебры алгебры M(GF(2),3).

2.2. Программа 2 Создадим массив порождающих троек подалгебр в пакете GAP

№	Текст программы	Комментарии
1	Al:=MatAlgebra(GF(2),3);	Построение алгебры матриц третьего порядка над полем GF(2)
2	El:=Elements(Al);	Создание массива элементов алгебры Al
3	Read("Dann.dan");	Считываем информацию файла (Результат выполнения первой программы)
4	A:=[];	Создаем массив
5	C:=[];	Создаем массив
6	for i in Gen do	Начало цикла
7	for j in Gen do	Начало цикла
8	for k in Gen do	Начало цикла
9	S:=Subalgebra(Al,[El[i],El[j],El[k]]);	Записывает подалгебру алгебры Al порожденную элементами El[i],El[j],El[k] в S
10	If Size(S) < 512 then	Если это не сама алгебра Al
11	el:=Elements(S);	Создание массива элементов подалгебры S
12	a:=Size(A);	Проверяем размер массива A
13	AddSet(A,el);	Заполняем массив A элементами из массива el
14	b:=Size(A);	Проверяем размер массива A
15	if b > a then	Проверяем наличие новых элементов

16	Add(C,[i,j,k]);	Если есть новые, то добавляем элемент алгебры Al в массив C
17	fi;	Конец условия
18	fi;	Конец условия
19	od;	Конец цикла
20	od;	Конец цикла
21	od;	Конец цикла
22	PrintTo("Dann3.dan","Gen-a-b-c:=",C,";");	Записываем полученные данные в файл

Замечание к выполнению программы:

Программа № 2 использует данные из программы № 1, поэтому файл "Dann.dan" желательно поместить в ту директорию, в которой будет запускаться программа № 2.

Результат выполнения (часть данных):

Gen3:=[[1, 139, 46], [1, 139, 49], [1, 139, 54], [1, 144, 1], [1, 144, 25], [1, 144, 30], [1, 144, 41], [1, 145, 1], [1, 145, 2], [1, 145, 3], [1, 145, 4], [1, 145, 5], [1, 145, 6], [1, 145, 7], [1, 145, 8], [1, 145, 9], [1, 145, 10], [1, 145, 15], [1, 145, 16], [1, 145, 46], [1, 145, 49], [1, 145, 50], [1, 145, 51], [1, 145, 52], [1, 145, 55], [1, 145, 56], [1, 145, 57], [1, 145, 58], [1, 145, 64], [1, 145, 65], [1, 145, 66], [1, 146, 1], [1, 146, 3], [1, 146, 4], [1, 146, 5], [1, 146, 6], [1, 146, 7], [1, 146, 8], [1, 146, 9], [1, 146, 10], [1, 146, 15], [1, 146, 49], [1, 146, 50], [1, 146, 51], [1, 146, 52], [1, 146, 54], [1, 146, 55], [1, 146, 56], [1, 146, 57], [1, 146, 58], [1, 146, 64], [1, 146, 65], [1, 146, 66], [1, 147, 1], [1, 147, 6], [1, 147, 7], [1, 147, 41], [1, 147, 46], [1, 147, 49], [1, 147, 52], [1, 147, 54], [1, 147, 55], [1, 147, 57], [1, 147, 66], [1, 148, 6], [1, 148, 50], [1, 149, 49], [1, 150, 1], [1, 151, 49], [1, 151, 65], [1, 152, 1], [1, 152, 9], [1, 152, 10], [1, 152, 15], [1, 152, 41], [1, 152, 46], [1, 152, 49], [1, 152, 50], [1, 152, 54], [1, 152, 55], [1, 152, 57], [1, 152, 64], [1, 152, 66], [1, 153, 1], [1, 153, 2], [1, 153, 49], [1, 154,

1], [1, 154, 8], [1, 154, 50], [1, 154, 55], [1, 154, 57], [1, 154, 64], [1, 155, 1], [1, 155, 6], [1, 155, 41], [1, 159, 1], [1, 159, 64], [1, 160, 1], [1, 161, 1], [1, 161, 2], [1, 161, 3], [1, 161, 4], [1, 161, 7], [1, 161, 8], [1, 161, 9], [1, 161, 10], [1, 162, 1], [1, 162, 3], [1, 162, 4], [1, 162, 7], [1, 162, 8], [1, 162, 9], [1, 162, 10], [1, 163, 1], [1, 163, 6], [1, 163, 7], [1, 164, 1], [1, 165, 1], [1, 165, 4], [1, 167, 1], [1, 168, 1], [1, 168, 25], [1, 169, 1], [1, 169, 8], [1, 169, 66], [1, 170, 1], [1, 171, 1], [1, 171, 66], [1, 174, 1], [1, 175, 1], [1, 176, 1], [1, 176, 22], [1, 176, 66], [1, 177, 1], [1, 177, 2], [1, 177, 3], [1, 177, 4], [1, 177, 9], [1, 177, 10], [1, 179, 1], [1, 179, 6], [1, 180, 1], [1, 181, 1], [1, 182, 1], [1, 183, 1], [1, 185, 1], [1, 185, 66], [1, 186, 1], [1, 187, 1], [1, 187, 6], [1, 187, 66], [1, 190, 1], [1, 190, 66], [1, 192, 1], [1, 192, 66], [1, 193, 1], [1, 193, 4], [1, 193, 5], [1, 193, 6], [1, 193, 10], [1, 193, 11], [1, 193, 14], [1, 193, 17], [1, 193, 18], [1, 193, 19], [1, 193, 22], [1, 193, 25], [1, 193, 28], [1, 193, 29], [1, 193, 30], [1, 193, 33], [1, 193, 37], [1, 193, 40], [1, 193, 41], [1, 193, 42], [1, 193, 46], [1, 193, 47], [1, 194, 17], [1, 194, 18], [1, 194, 41], [1, 196, 1], [1, 196, 6], [1, 196, 8], [1, 196, 10], [1, 196, 11], [1, 196, 14], [1, 196, 15], [1, 196, 18], [1, 196, 19], [1, 196, 22], [1, 196, 25], [1, 196, 28], [1, 196, 29], [1, 196, 30], [1, 196, 33], [1, 196, 37], [1, 196, 41], [1, 196, 43], [1, 196, 46], [1, 196, 47], [1, 196, 49], [1, 196, 54], [1, 196, 55], [1, 196, 57], [1, 196, 64], [1, 197, 1], [1, 197, 10], [1, 197, 15], [1, 197, 19], [1, 197, 22], [1, 197, 28], [1, 197, 43], [1, 197, 46], [1, 197, 57], [1, 197, 64], [1, 198, 1], [1, 198, 31], [1, 199, 1], [1, 199, 17], [1, 199, 22], [1, 199, 30], [1, 199, 41], [1, 199, 46], [1, 199, 57], [1, 199, 62], [1, 200, 1], [1, 200, 10], [1, 200, 19], [1, 200, 28], [1, 202, 18], [1, 202, 19], [1, 202, 37], [1, 202, 40], [1, 203, 1], [1, 203, 8], [1, 203, 37], [1, 203, 46], [1, 203, 47], [1, 206, 1], [1, 206, 28], [1, 207, 1], [1, 210, 1], [1, 210, 8], [1, 210, 10], [1, 210, 15], [1, 210, 46], [1, 210, 50], [1, 210, 55], [1, 210, 57],[217, 274, 1], [217, 274, 55], [217, 274, 64], [217, 279, 1], [217, 281, 1], [217, 281, 64], [217, 288, 1], [217, 289, 1], [217, 296, 1], [217, 345, 1], [217, 361, 1], [220, 266, 1]];

Каждая из полученных выше троек номеров порождает подалгебру, содержащую 2^k элементов, где k э {0, 1, 2, 3, 4, 5, 6, 7}. Известно, что алгебра матриц *M3(GF(2))* не содержит подалгебр порядка 256. Все подалгебры попарно различны, но некоторые могут быть изоморфны между собой.

2.3. Программа 3 Программа построения подалгебры, ее решетки и типа

№	Текст программы	Комментарии
1	GetData:=function(x,y,z);	Задаем функцию
2	local A, El, i, i1, j, k, a, b, c, s, S, sub, tip, pokr, sl, el, l, l1, m, m1, n, n1;	Задаем локальные переменные
3	A:=MatAlgebra(GF(2),3);	Построение алгебры матриц третьего порядка над полем GF(2)
4	El:=Elements(A);	Создание массива элементов алгебры A
5	S:=Subalgebra(A,[El[x],El[y],El[z]]);	Записывает подалгебру алгебры A порожденную элементами El[x],El[y],El[z] в S
6	sub:=[];	Создание массива
7	tip:=[];	Создание массива для хранения типа
8	for a in S do	Начало цикла
9	for b in S do	Начало цикла
10	for c in S do	Начало цикла
11	s:=Subalgebra(A,[a,b,c]);	Записывает подалгебру алгебры A порожденную элементами a,b,c в s
12	AddSet(sub, Elements(s));	Записываем элементы подалгебры s в массив sub
13	od; od; od;	Конец циклов
14	for j in [1..Size(sub)] do	Начало цикла

15	Add(tip,Size(sub[j]));	Записываем размеры всех элементов подалгебры s в массив tip
16	od;	Конец циклов
17	tip:=Collected(tip);	Преобразуем массив, подсчитываем количество одинаковых элементов
18	sub:=[];	Создание массива
19	for i1 in [1..9] do	Начало цикла
20	sub[i1]:=[];	Создание массива в массиве
21	od;	Конец циклов
22	pokr:=[];	Создание массива, для хранения покрытия
23	for i in S do	Начало цикла
24	for k in S do	Начало цикла
25	for j in S do	Начало цикла
26	s1:=Subalgebra(S,[i,k,j]);	Записывает подалгебру алгебры A порожденную элементами i, k, j в `s1`
27	if Size(s1)=1 then	Начало условия
28	AddSet(sub[1],Elements(s1));	Если размерность подалгебры равна 1, то записываем в подмассив с индексом 1
29	fi;	Конец условия
30	if Size(s1)=2 then	Начало условия
31	AddSet(sub[2],Elements(s1));	Аналогично
32	fi;	Конец условия
33	if Size(s1)=4 then	Начало условия
34	AddSet(sub[3],Elements(s1));	Аналогично
35	fi;	Конец условия

36	if Size(s1)=8 then	Начало условия
37	AddSet(sub[4],Elements(s1));	Аналогично
38	fi;	Конец условия
39	if Size(s1)=16 then	Начало условия
40	AddSet(sub[5],Elements(s1));	Аналогично
41	fi;	Конец условия
42	if Size(s1)=32 then	Начало условия
43	AddSet(sub[6],Elements(s1));	Аналогично
44	fi;	Конец условия
45	if Size(s1)=64 then	Начало условия
46	AddSet(sub[7],Elements(s1));	Аналогично
47	fi;	Конец условия
48	if Size(s1)=128 then	Начало условия
49	AddSet(sub[8],Elements(s1));	Аналогично
50	fi;	Конец условия
51	if Size(s1)=512 then	Начало условия
52	AddSet(sub[9],Elements(s1));	Аналогично
53	fi;	Конец условия
54	od; od; od;	Конец циклов
55	for m1 in [1..Size(sub)] do	Начало цикла
56	for l1 in [1..Size(sub[m1])] do	Начало цикла
57	for n1 in [1..Size(sub[m1+1])] do	Начало цикла
58	if IsSubset(sub[m1+1][n1], sub[m1][l1]) = true	Начало условия

59	then Add(pokr,[[m1,l1],[m1+1,n1]]);	Если элемент из подалгебры большего размера принадлежит подалгебре меньшего размера, то добавляем связь между подалгебрами
60	fi;	Конец условия
61	od; od; od;	Конец циклов
62	PrintTo("Data00.txt","x= ",x,";", " y= ",y,";"," z= ",z,";","\nType= ",tip, "\nPokr= ", pokr, "\n");	Записываем полученный результат в файл (порождающая тройка, тип данных и покрытие)
63	end;	Конец функции

Результат выполнения (на примере образующей тройке [1, 193, 5]):

x= 1; y= 193; z= 5;

Type= [[1, 1], [2, 10], [4, 13], [8, 3], [16, 1]]

Pokr= [[[1, 1], [2, 1]], [[1, 1], [2, 2]], [[1, 1], [2, 3]], [[1, 1], [2, 4]], [[1, 1], [2, 5]], [[1, 1], [2, 6]], [[1, 1], [2, 7]], [[1, 1], [2, 8]], [[1, 1], [2, 9]], [[1, 1], [2, 10]], [[2, 1], [3, 1]], [[2, 1], [3, 2]], [[2, 1], [3, 3]], [[2, 2], [3, 1]], [[2, 2], [3, 4]], [[2, 2], [3, 5]], [[2, 3], [3, 1]], [[2, 3], [3, 6]], [[2, 3], [3, 7]], [[2, 4], [3, 2]], [[2, 4], [3, 8]], [[2, 4], [3, 9]], [[2, 5], [3, 2]], [[2, 5], [3, 10]], [[2, 5], [3, 11]], [[2, 6], [3, 3]], [[2, 6], [3, 4]], [[2, 6], [3, 8]], [[2, 7], [3, 3]], [[2, 7], [3, 5]], [[2, 7], [3, 6]], [[2, 7], [3, 9]], [[2, 7], [3, 10]], [[2, 7], [3, 12]], [[2, 7], [3, 13]], [[2, 8], [3, 4]], [[2, 8], [3, 6]], [[2, 8], [3, 11]], [[2, 9], [3, 7]], [[2, 9], [3, 8]], [[2, 9], [3, 10]], [[2, 10], [3, 7]], [[2, 10], [3, 11]], [[2, 10], [3, 12]], [[3, 1], [4, 1]], [[3, 2], [4, 2]], [[3, 3], [4, 1]], [[3, 3], [4, 2]], [[3, 4], [4, 1]], [[3, 5], [4, 1]], [[3, 6], [4, 1]], [[3, 6], [4, 3]], [[3, 7], [4, 3]], [[3, 8], [4, 2]], [[3, 9], [4, 2]], [[3, 10], [4, 2]], [[3, 10], [4, 3]], [[3, 11], [4, 3]], [[3, 12], [4, 3]], [[4, 1], [5, 1]], [[4, 2], [5, 1]], [[4, 3], [5, 1]]].

2.4. Построение диаграмм

Используя полученные данные от выполнения программы 3, строится диаграмма решетки подалгебр алгебры *S*. Построение осуществляется в несколько этапов:

1. Изображаются подалгебры алгебры *S* точками (или кружочками).
2. Изображается отношение покрытия, соединяя покрываемый элемент с покрывающим отрезком.

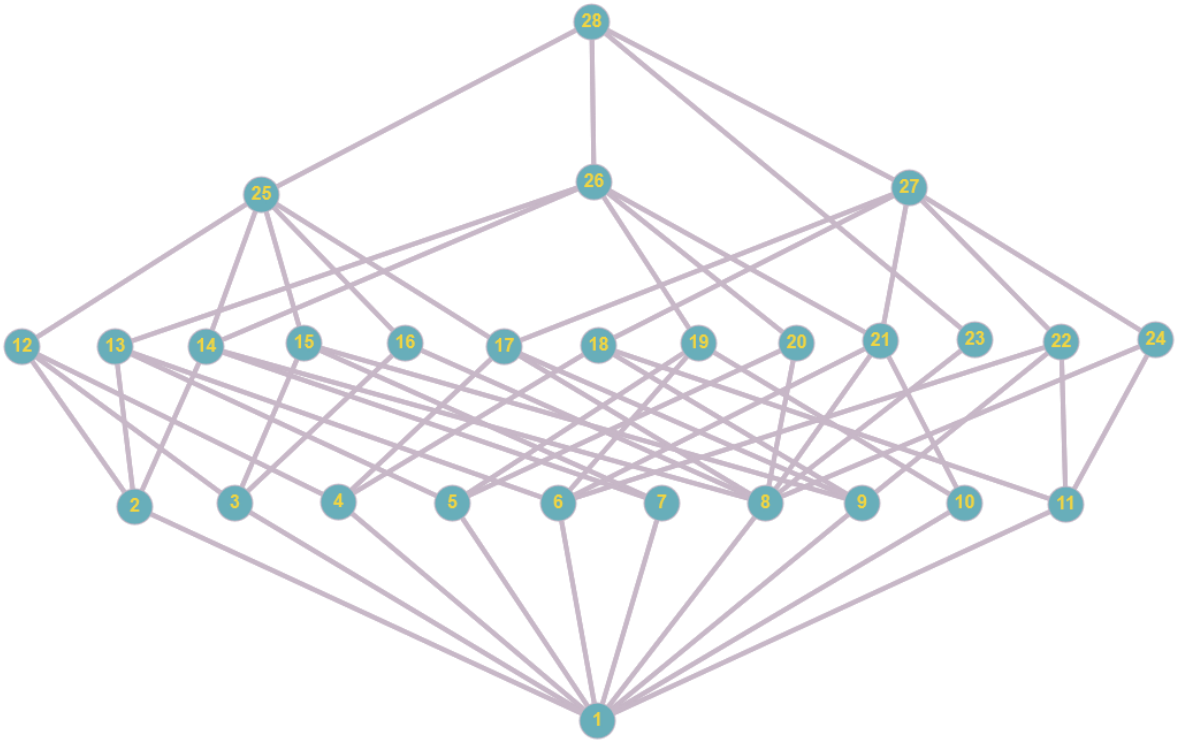

Рисунок 1. Тип [1, 10, 13, 3, 1]

ГЛАВА III. Алгоритмы создания программы для построения трехмерной модели решетки подалгебры алгебры M(GF(2),3)

Интерфейс программы:

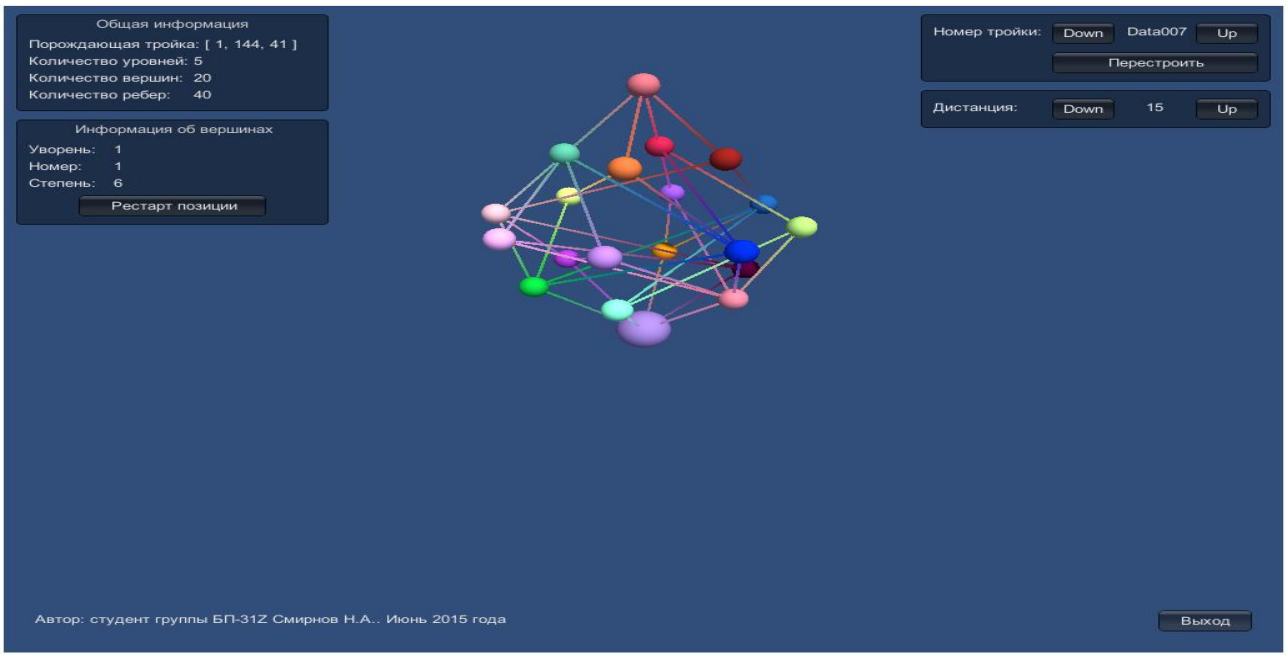

Рис 3.1 – Общий вид программы

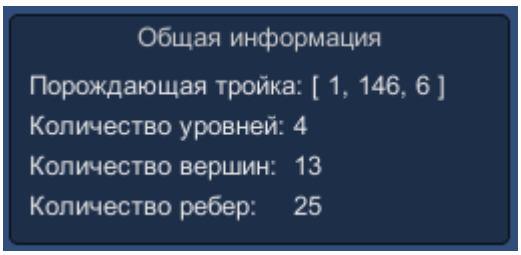

Рис 3.2 – Окно «Общая информация»

Окно «Общая информация» содержит информацию о текущем отображаемом графе.

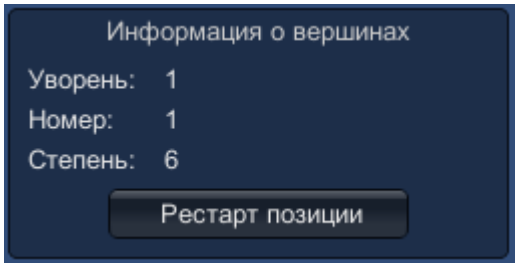

Рис 3.3. – Окно «Информация о вершинах»

Окно «Информация о вершинах» содержит информацию о текущей выбранной вершине графа. Кнопка «Рестарт позиции» возвращает текущую выбранную вершину графа в начальное положение (после перемещением ее курсором мыши).

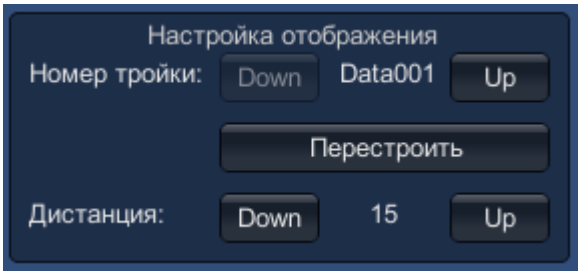

Рис 3.3. – Окно «Настройки отображения»

Окно «Настройки отображения» содержит настройки сцены: настройки дистанции до графа и выбор нужного графа для построения, также если не менять номер тройки(графа) в случае изменения позиций вершин нажав «Перестроить» все вершины встанут на свои места.

Чтобы закрыть программу необходимо нажать кнопку «Закрыть»;

Возможности программы:

1) Построения графов из образующих троек,
2) Отображение информации о графе и выбранной вершины,
3) Обзор графа в 3D и перемещение вершин.

Описание программы

Рассмотрим первую порождающую тройку - (1, 139, 46). Запустив скрипт «Zapusk3.g» мы сформировали файл «Data001.txt». Для обработки 3D покетом и построения графи содержимое файла нас не устраивает:

x= 1; y= 139; z= 46;

Type= [[1, 1], [2, 8], [4, 12], [8, 6], [16, 1]]

Pokr= [[[1, 1], [2, 1]], [[1, 1], [2, 2]], [[1, 1], [2, 3]], [[1, 1], [2, 4]],

[[1, 1], [2, 5]], [[1, 1], [2, 6]], [[1, 1], [2, 7]], [[1, 1], [2, 8]],
[[2, 1], [3, 1]], [[2, 1], [3, 2]], [[2, 1], [3, 3]], [[2, 2], [3, 4]],
[[2, 2], [3, 5]], [[2, 2], [3, 6]], [[2, 3], [3, 1]], [[2, 3], [3, 4]],
[[2, 3], [3, 9]], [[2, 3], [3, 10]], [[2, 3], [3, 11]], [[2, 3], [3, 12]],
[[2, 4], [3, 2]], [[2, 4], [3, 5]], [[2, 4], [3, 7]], [[2, 4], [3, 8]],
[[2, 4], [3, 9]], [[2, 5], [3, 2]], [[2, 5], [3, 6]], [[2, 5], [3, 10]],
[[2, 6], [3, 3]], [[2, 6], [3, 5]], [[2, 6], [3, 11]], [[2, 7], [3, 3]],
[[2, 7], [3, 6]], [[2, 7], [3, 7]], [[2, 7], [3, 12]], [[2, 8], [3, 8]],
[[2, 8], [3, 12]], [[3, 1], [4, 2]], [[3, 1], [4, 3]], [[3, 2], [4, 1]],
[[3, 2], [4, 2]], [[3, 3], [4, 1]], [[3, 3], [4, 3]], [[3, 4], [4, 4]],
[[3, 4], [4, 5]], [[3, 5], [4, 1]], [[3, 5], [4, 4]], [[3, 6], [4, 1]],
[[3, 6], [4, 5]], [[3, 7], [4, 1]], [[3, 7], [4, 6]], [[3, 8], [4, 6]],
[[3, 9], [4, 2]], [[3, 9], [4, 4]], [[3, 9], [4, 6]], [[3, 10], [4, 2]],
[[3, 10], [4, 5]], [[3, 11], [4, 3]], [[3, 11], [4, 4]], [[3, 12], [4, 3]],
[[3, 12], [4, 5]], [[3, 12], [4, 6]], [[4, 1], [5, 1]], [[4, 2], [5, 1]],
[[4, 3], [5, 1]], [[4, 4], [5, 1]], [[4, 5], [5, 1]], [[4, 6], [5, 1]]]

Загружаем строку в программу и форматируем ей в желаемый для нас вид.

```
#region GetData
/// <summary>
/// Приводим строку из файла к нужному нам виду
/// </summary>
/// <param name="m_ID">ID файла</param>
/// <returns>Отформатированную строку, готовую к дальнейшим действиям над ней</returns>
private string GetData(int m_ID)
{
    string text = File.ReadAllText(file_list[m_ID]);
    text = Regex.Replace(text, @"\s+", "");
```

```
            text = text.Insert(text.IndexOf("y"), "\n");
            text = text.Insert(text.IndexOf("z"), "\n");
            text = text.Insert(text.IndexOf("Type"), "\n");
            text = text.Insert(text.IndexOf("Pokr"), "\n");
            return text;
        }
        #endregion
```

Разделив строку на интересующие нам абзацы получим:

x=1;

y=139;

z=46;

Type=[[1,1],[2,8],[4,12],[8,6],[16,1]]

Pokr=[[[1,1],[2,1]],[[1,1],[2,2]],[[1,1],[2,3]],[[1,1],[2,4]],[[1,1],[2,5]],[[1,1],[2,6]],[[1,1],[2,7]],[[1,1],[2,8]],[[2,1],[3,1]],[[2,1],[3,2]],[[2,1],[3,3]],[[2,2],[3,4]],[[2,2],[3,5]],[[2,2],[3,6]],[[2,3],[3,1]],[[2,3],[3,4]],[[2,3],[3,9]],[[2,3],[3,10]],[[2,3],[3,11]],[[2,3],[3,12]],[[2,4],[3,2]],[[2,4],[3,5]],[[2,4],[3,7]],[[2,4],[3,8]],[[2,4],[3,9]],[[2,5],[3,2]],[[2,5],[3,6]],[[2,5],[3,10]],[[2,6],[3,3]],[[2,6],[3,5]],[[2,6],[3,11]],[[2,7],[3,3]],[[2,7],[3,6]],[[2,7],[3,7]],[[2,7],[3,12]],[[2,8],[3,8]],[[2,8],[3,12]],[[3,1],[4,2]],[[3,1],[4,3]],[[3,2],[4,1]],[[3,2],[4,2]],[[3,3],[4,1]],[[3,3],[4,3]],[[3,4],[4,4]],[[3,4],[4,5]],[[3,5],[4,1]],[[3,5],[4,4]],[[3,6],[4,1]],[[3,6],[4,5]],[[3,7],[4,1]],[[3,7],[4,6]],[[3,8],[4,6]],[[3,9],[4,2]],[[3,9],[4,4]],[[3,9],[4,6]],[[3,10],[4,2]],[[3,10],[4,5]],[[3,11],[4,3]],[[3,11],[4,4]],[[3,12],[4,3]],[[3,12],[4,5]],[[3,12],[4,6]],[[4,1],[5,1]],[[4,2],[5,1]],[[4,3],[5,1]],[[4,4],[5,1]],[[4,5],[5,1]],[[4,6],[5,1]]]

Для дальнейших действий зададим переменные для хранения данных:

```
    // Текущий ID порождающей тройки
    public int ID = 0;
    // Ссылка на прифабы вершина/ребро
```

```
public Object PrefabVertex;
public Object PrefabEage;
// Список ссылок на файли с данными порождающих троек
public string[] file_list;
// Переменная для хранения информации о текущей тройке
string text = string.Empty;
// Тройкаи
public Vector3 MatrixIndex;
// Переменная стартовой позиции для расчета позиции вершин
public Vector3 VertexPosition;
// Список для хранения типа алгебры
public List<int[]> m_type = new List<int[]>();
// Список для хранения покрытия
public List<int[,]> m_pokr = new List<int[,]>();
// Дистанция между уровняли
public float m_Distance = 3f;
// Радиус окружности уровня на котором размещаются вершины
public float m_Radius = 2;
// Размещаем вершины равномерно по всей окружности
public float Angle = 360;
```

Чтобы программа не выдавала ошибок на префабы опишем функцию Awake() {} для назначению переменной соответствующего объекта:

```
// Выполняется при первом обращение к объекту
void Awake()
{
   PrefabVertex = Resources.Load("Prefabs/Vertex");
   PrefabEage = Resources.Load("Prefabs/Eage");
}
```

Чтобы погружать ресурсы на сцену через скрипты необходимо в главном каталоге проекта создаль папку «Resources»:

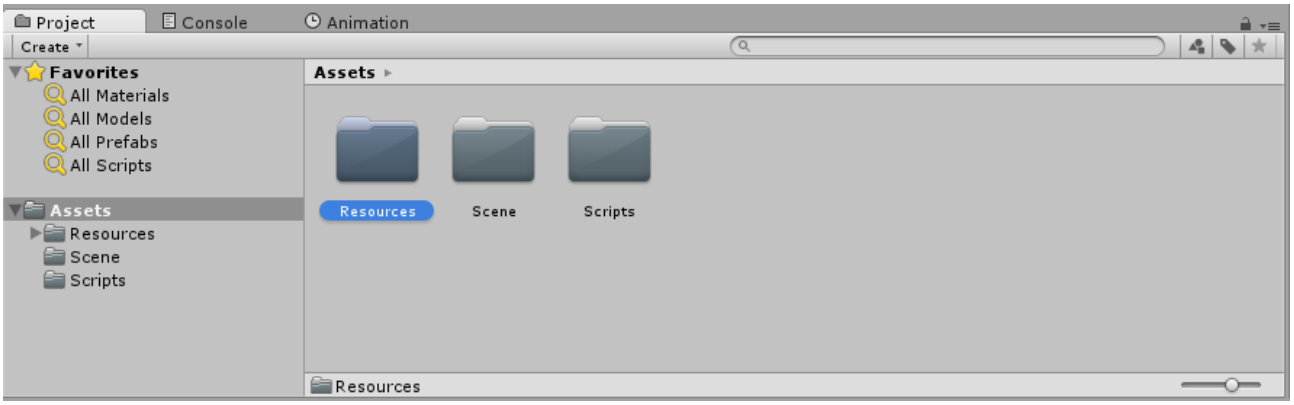

Рис. 3.4 – Создание папки «Resources»

При первом старте программы мы формируем список файлов с данными исследуемых порождающих троек, и производим инициализацию (создание) графа:

```
// Выполняется при первом объявлении объекта
void Start ()
{
   // Возвращает имена файлов из указанного каталога, отвечающие условиям заданного шаблона поиска
   file_list = Directory.GetFiles(@"C:\Files\GAP", "*.txt");
   // Построение графа
   InstantiateGraph(ID);
}
```

Функция «InstantiateGraph» запускает в себе следующие функциии:

```
        text = GetData(UID);              // Приводим строку из файла к нужному нам
виду
        GetMatrixIndex(text);     // Извлекаем образующую тройку
        GetType(text);            // Извлекаем тип диаграммы
        GetPokr(text);            // Извлекаем покрытие
        CreateVertex();           // Создаем Vertex
        CreateEage();                     // Создаем Eage


    #region CreateVertex
    /// <summary>
    /// Создаем Vertex
    /// </summary>
    private void CreateVertex()
    {
        float m_Angle = Angle * Mathf.Deg2Rad;
        for (int i = 0; i < m_type.Count; i++)
        {
            for (int j = 0; j < m_type[i][1]; j++)
            {
                switch (i)
                {
                    case 0:
                        VertexPosition = new Vector3();
                        break;
                    default:
                        float _z = 0;
                        float _x = 0;
                        if (m_type[i][1] != 1)
```

```
                    {
                        m_Radius = m_type[i][1] / Mathf.PI;
                        _x = transform.position.x + Mathf.Sin(m_Angle / m_type[i][1]
* j) * m_Radius;
                        _z = transform.position.z + Mathf.Cos(m_Angle /
m_type[i][1] * j) * m_Radius;
                    }
                    VertexPosition.x = _x;
                    VertexPosition.y = m_Distance * i;
                    VertexPosition.z = _z;
                    break;
                }
                GameObject go = Instantiate(PrefabVertex, VertexPosition, Quater-
nion.identity) as GameObject;
                go.transform.SetParent(gameObject.transform);
                go.GetComponent<Vertex>().ID = i + 1;
                go.GetComponent<Vertex>().NUMBER = j + 1;
                if (i == 0 && j == 0)
                {
                    go.transform.localScale = new Vector3(1, 1, 1);
                    GetComponent<GUIManager>().m_SelectedObject = go;
                }
            }
        }
    }
    #endregion

    #region GetPokr
    /// <summary>
```

```
        /// Извлекаем покрытие
        /// </summary>
        /// <param name="text">Отформатированный текст</param>
        private void GetPokr(string text)
        {
            m_pokr.Clear();
            string pokr = text.Split('\n')[4].Replace("Pokr=", "").Replace("[", "").Replace("]", "");
            for (int i = 0; i < pokr.Split(',').Length / 4; i++)
            {
                int[,] temp = new int[2, 2];
                int j = 0;
                for (int k = 0; k < 2; k++)
                    for (int q = 0; q < 2; q++)
                    {
                        temp[k, q] = int.Parse(pokr.Split(',')[j + i * 4]);
                        j++;
                    }
                m_pokr.Add(temp);
            }
        }
        #endregion

        #region GetType
        /// <summary>
        /// Извлекаем тип диаграммы
        /// </summary>
        /// <param name="text">Отформатированный текст</param>
        private void GetType(string text)
```

```
        {
            m_type.Clear();
            string type = text.Split('\n')[3].Replace("Type=", "").Replace("[", "").Replace("]", "");
            for (int i = 0; i < type.Split(',').Length / 2; i++)
            {
                int[] temp = new int[2];
                for (int j = 0; j < 2; j++)
                    temp[j] = int.Parse(type.Split(',')[j + i * 2]);
                m_type.Add(temp);
            }
        }
        #endregion

        #region GetMatrixIndex
        /// <summary>
        /// Извлекаем образующую тройку
        /// </summary>
        /// <param name="text">Отформатированный текст</param>
        private void GetMatrixIndex(string text)
        {
            MatrixIndex = new Vector3(
                float.Parse(text.Split('\n')[0].Substring(text.Split('\n')[0].IndexOf("=") + 1, text.Split('\n')[0].Length - 3)),
                float.Parse(text.Split('\n')[1].Substring(text.Split('\n')[1].IndexOf("=") + 1, text.Split('\n')[1].Length - 3)),
                float.Parse(text.Split('\n')[2].Substring(text.Split('\n')[2].IndexOf("=") + 1, text.Split('\n')[2].Length - 3))
                );
```

```
        }
        #endregion

        #region GetData
        /// <summary>
        /// Приводим строку из файла к нужному нам виду
        /// </summary>
        /// <param name="m_ID">ID файла</param>
        /// <returns>Отформатированную строку готовую к дальнейшим действиям над ней</returns>
        private string GetData(int m_ID)
        {
            string text = File.ReadAllText(file_list[m_ID]);
            text = Regex.Replace(text, @"\s+", "");
            text = text.Insert(text.IndexOf("y"), "\n");
            text = text.Insert(text.IndexOf("z"), "\n");
            text = text.Insert(text.IndexOf("Type"), "\n");
            text = text.Insert(text.IndexOf("Pokr"), "\n");
            return text;
        }
        #endregion
```

Скрипт для управления камерой и GUI изменения дистанции до графа:

```
public class MainCamera : MonoBehaviour {

public Transform target;

public float Distance = 15.0f;
```

```
public float XSpeed = 10.0f;
public float YSpeed = 50.0f;

public float YMinLimit = 10.0f;
public float YmaxLimit = 80.0f;

public float DistanceMin = 3.0f;
public float DistanceMax = 35.0f;

private float X = 0.0f;
private float Y = 0.0f;

   void Awake()
   {
      target = GameObject.FindGameObjectWithTag("Graph").transform;
   }

void Start () {
        var angles = transform.eulerAngles;
        X = angles.x;
        Y = angles.y;

        if(GetComponent<Rigidbody>()){
                GetComponent<Rigidbody>().freezeRotation = false;
        }
}

void LateUpdate(){
        if (Input.GetKey (KeyCode.Mouse1)) {
```

```
                                        Cursor.visible = false;
                                        X += Input.GetAxis ("Mouse X") * XSpeed *
Distance * 0.02f;
                                        Y -= Input.GetAxis ("Mouse Y") * YSpeed *
0.02f;
                        } else
                                        Cursor.visible = true;
                if (Input.touchCount > 0 && Input.GetTouch (0).phase == Touch-
Phase.Moved) {
                                        X += Input.GetTouch (0).deltaPosition.x *
XSpeed/5 * Distance * 0.02f;
                                        Y -= Input.GetTouch (0).deltaPosition.y *
YSpeed/5 * 0.02f;
                        }
                Y = ClampAngle (Y, YMinLimit, YmaxLimit);
                Quaternion Rotation = Quaternion.Euler (Y, X, 0);
                Vector3 Position = Rotation * new Vector3 (0.0f, 0.0f, - Distance) + tar-
get.position;

                transform.rotation = Rotation;
                transform.position = Position;
        }

        static float ClampAngle(float angle, float min, float max){
                if (angle < -360.0f)
                                        angle = +360.0f;
                if (angle > 360.0f)
                                        angle -= 360.0f;
                return Mathf.Clamp (angle, min, max);
```

```
}

    void OnGUI()
    {
        GUI.Label(new Rect(Screen.width - 280, 100, 100, 25), "Дистанция:");
        if (Distance == DistanceMin) GUI.enabled = false;
        if (GUI.Button(new Rect(Screen.width - 185, 100, 50, 25), "Down"))
            Distance--;
        GUI.enabled = true;
        GUI.Label(new Rect(Screen.width - 110, 100, 50, 25), Distance.ToString());
        if (Distance == DistanceMax) GUI.enabled = false;
        if (GUI.Button(new Rect(Screen.width - 70, 100, 50, 25), "Up"))
            Distance++;
        GUI.enabled = true;
    }

    void Update()
    {
        if (Distance < DistanceMin) Distance = DistanceMin;
        if (Distance > DistanceMax) Distance = DistanceMax;
    }
}
```

Скрипт управления ребрами графа:

```
public class Eage : MonoBehaviour {
    // номер уровня в котором находится начальная вершина
    public int StartID;
    // порядковый номер начальной вершины
```

```
    public int StartNUMBER;
    // номер уровня в котором находится конечном вершина
    public int EndID;
    // порядковый номер конечной вершины
    public int EndNUMBER;
    // Объект рисования ребра графа
    public LineRenderer lineRenderer;
    // Цвета вершин (соответственно)
    public Color ColorStart, ColorEnd;
    void Awake()
    {
      lineRenderer = GetComponent<LineRenderer>();
    }

  void Start ()
    {
      foreach (GameObject go in GameObject.FindGameObjectsWithTag("Vertex"))
      {
        if (go.GetComponent<Vertex>().ID == StartID && go.GetComponent<Vertex>().NUMBER == StartNUMBER)
          go.GetComponent<Vertex>().Begree++;
        if (go.GetComponent<Vertex>().ID == EndID && go.GetComponent<Vertex>().NUMBER == EndNUMBER)
          go.GetComponent<Vertex>().Begree++;
      }
  }

  void Update ()
```

```
        {
            foreach (GameObject go in GameObject.FindGameObjectsWithTag("Vertex"))
            {
                if (go.GetComponent<Vertex>().ID == StartID && go.GetComponent<Vertex>().NUMBER == StartNUMBER)
                    ColorStart = go.GetComponent<MeshRenderer>().materials[0].color;
                if (go.GetComponent<Vertex>().ID == EndID && go.GetComponent<Vertex>().NUMBER == EndNUMBER)
                    ColorEnd = go.GetComponent<MeshRenderer>().materials[0].color;
            }
            lineRenderer.SetColors(ColorStart, ColorEnd);
            foreach(GameObject go in GameObject.FindGameObjectsWithTag("Vertex"))
            {
                if(go.GetComponent<Vertex>().ID == StartID && go.GetComponent<Vertex>().NUMBER == StartNUMBER)
                    lineRenderer.SetPosition(0, go.transform.position);
                if (go.GetComponent<Vertex>().ID == EndID && go.GetComponent<Vertex>().NUMBER == EndNUMBER)
                    lineRenderer.SetPosition(1, go.transform.position);
            }
        }
        }
```

Скрипт управления вершинами:

```
public class Vertex : MonoBehaviour
{
    // Уровень
```

```
public int ID;
// Порядковый номер
public int NUMBER;
// Степень вершины
public int Begree;
// Начальная позиция
public Vector3 mainPosition;
// Начальный размер
private Vector3 m_Normal = new Vector3(.5f, .5f, .5f);
// Размер выделеной вершины
private Vector3 m_Select = new Vector3(1, 1, 1);
GUIManager gui;

void Awake()
{
   mainPosition = transform.position;
   gui = GameObject.FindGameObjectWithTag("Graph").GetCompo-
nent<GUIManager>();
}

void Start()
{
   GetComponent<Renderer>().material.color = new Color32((byte)Ran-
dom.Range(0, 255), (byte)Random.Range(0, 255), (byte)Random.Range(0, 255),
255);
}

private Vector3 screenPoint;
private Vector3 offset;
```

```
void OnMouseDown()
{
  screenPoint = Camera.main.WorldToScreenPoint(gameObject.transform.position);
  offset = gameObject.transform.position - Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenPoint.z));
  if (gui.m_SelectedObject != gameObject)
    if (GetComponent<Renderer>() != null)
    {
      GetComponent<Renderer>().material.color = new Color32((byte)Random.Range(0, 255), (byte)Random.Range(0, 255), (byte)Random.Range(0, 255), 255);
      if (gui.m_SelectedObject != null) gui.m_SelectedObject.transform.localScale = m_Normal;
      transform.localScale = m_Select;
      gui.m_SelectedObject = gameObject;
    }
}

void OnMouseDrag()
{
  Vector3 curScreenPoint = new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenPoint.z);
  Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenPoint) + offset;
  transform.position = curPosition;
}}
```

Заключение

В ходе работы мы разработали алгоритм и программы для исследования решеток подалгебр алгебр малых размерностей в прикладном пакете *GAP*, для запуска необходимо набрать в среде *GAP* команду: *GetData(x, y, z)*, где *x, y* и *z* – номера порождающих элементов подалгебры.

Запустить разработанные алгоритмы и программу в пакете *Unity3D* можно двумя способами:

1) Запуск приложения внутри пакета *Unity3D* нажав на кнопку *«Play»* на панели инструментов.
2) Нажав сочетание клавиш *«Ctrl + Shift + B»*, появится диалог, в котором предложат указать место сохранения программы. После указания места *Unity3D* скомпилирует программу и запустит ее автоматически.

В ходе работы мы решили все поставленные задачи. Цель работы достигнута.

Литература

1. Биркгоф, Г. Теория решеток; пер. с англ. В. Н. Салий под ред. Л. А. Скорнякова. – М.: Наука, 1984. – 568 с.
2. Гретцер, Г. Общая теория решеток; пер. с англ. А. Д. Больбота, В. А. Горбунова, В. И. Туманова под ред. Д. М. Смирнова. – М. : Мир, 1982. – 456 с.
3. Коробков С. С. Введение в теорию решеток: Учеб. пособие по спец. курсу. Урал. гос. пед. ун-т. — Екатеринбург: Б.и., 1996. – 64с.
4. Коробков С.С. Вычисления в матричных алгебрах (Прикладные аспекты алгебры и информатики). (Рукопись). Екатеринбург, 2014.
5. Система компьютерной алгебры GAP: http://www.gap-system.org/
6. GAP Manual: http://www.gap-system.org/Doc/manuals.html
7. Бесплатное ПО от Microsoft «Visual Studio Community 2015»: https://www.microsoft.com/ru-ru/SoftMicrosoft/vs2015Community.aspx
8. Беспланое ПО, мощный графический движок и редактор Unity: https://unity3d.com/ru
9. Graph Online: http://graphonline.ru/

Приложение

Массив матриц алгебры $A = M_3(GF(2))$

$$m_1=\begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}, m_2=\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}, m_3=\begin{pmatrix}0&0&0\\0&0&0\\0&1&0\end{pmatrix}, m_4=\begin{pmatrix}0&0&0\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_5=\begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}, m_6=\begin{pmatrix}0&0&0\\0&0&0\\1&0&1\end{pmatrix}, m_7=\begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix}, m_8=\begin{pmatrix}0&0&0\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_9=\begin{pmatrix}0&0&0\\0&0&1\\0&0&0\end{pmatrix}, m_{10}=\begin{pmatrix}0&0&0\\0&0&1\\0&0&1\end{pmatrix}, m_{11}=\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix}, m_{12}=\begin{pmatrix}0&0&0\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{13}=\begin{pmatrix}0&0&0\\0&0&1\\1&0&0\end{pmatrix}, m_{14}=\begin{pmatrix}0&0&0\\0&0&1\\1&0&1\end{pmatrix}, m_{15}=\begin{pmatrix}0&0&0\\0&0&1\\1&1&0\end{pmatrix}, m_{16}=\begin{pmatrix}0&0&0\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{17}=\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}, m_{18}=\begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix}, m_{19}=\begin{pmatrix}0&0&0\\0&1&0\\0&1&0\end{pmatrix}, m_{20}=\begin{pmatrix}0&0&0\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{21}=\begin{pmatrix}0&0&0\\0&1&0\\1&0&0\end{pmatrix}, m_{22}=\begin{pmatrix}0&0&0\\0&1&0\\1&0&1\end{pmatrix}, m_{23}=\begin{pmatrix}0&0&0\\0&1&0\\1&1&0\end{pmatrix}, m_{24}=\begin{pmatrix}0&0&0\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{25}=\begin{pmatrix}0&0&0\\0&1&1\\0&0&0\end{pmatrix}, m_{26}=\begin{pmatrix}0&0&0\\0&1&1\\0&0&1\end{pmatrix}, m_{27}=\begin{pmatrix}0&0&0\\0&1&1\\0&1&0\end{pmatrix}, m_{28}=\begin{pmatrix}0&0&0\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{29}=\begin{pmatrix}0&0&0\\0&1&1\\1&0&0\end{pmatrix}, m_{30}=\begin{pmatrix}0&0&0\\0&1&1\\1&0&1\end{pmatrix}, m_{31}=\begin{pmatrix}0&0&0\\0&1&1\\1&1&0\end{pmatrix}, m_{32}=\begin{pmatrix}0&0&0\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{33}=\begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix}, m_{34}=\begin{pmatrix}0&0&0\\1&0&0\\0&0&1\end{pmatrix}, m_{35}=\begin{pmatrix}0&0&0\\1&0&0\\0&1&0\end{pmatrix}, m_{36}=\begin{pmatrix}0&0&0\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{37}=\begin{pmatrix}0&0&0\\1&0&0\\1&0&0\end{pmatrix}, m_{38}=\begin{pmatrix}0&0&0\\1&0&0\\1&0&1\end{pmatrix}, m_{39}=\begin{pmatrix}0&0&0\\1&0&0\\1&1&0\end{pmatrix}, m_{40}=\begin{pmatrix}0&0&0\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{41}=\begin{pmatrix}0&0&0\\1&0&1\\0&0&0\end{pmatrix}, m_{42}=\begin{pmatrix}0&0&0\\1&0&1\\0&0&1\end{pmatrix}, m_{43}=\begin{pmatrix}0&0&0\\1&0&1\\0&1&0\end{pmatrix}, m_{44}=\begin{pmatrix}0&0&0\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{45}=\begin{pmatrix}0&0&0\\1&0&1\\1&0&0\end{pmatrix}, m_{46}=\begin{pmatrix}0&0&0\\1&0&1\\1&0&1\end{pmatrix}, m_{47}=\begin{pmatrix}0&0&0\\1&0&1\\1&1&0\end{pmatrix}, m_{48}=\begin{pmatrix}0&0&0\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{49}=\begin{pmatrix}0&0&0\\1&1&0\\0&0&0\end{pmatrix}, m_{50}=\begin{pmatrix}0&0&0\\1&1&0\\0&0&1\end{pmatrix}, m_{51}=\begin{pmatrix}0&0&0\\1&1&0\\0&1&0\end{pmatrix}, m_{52}=\begin{pmatrix}0&0&0\\1&1&0\\0&1&1\end{pmatrix},$$

$$m_{53}=\begin{pmatrix}0&0&0\\1&1&0\\1&0&0\end{pmatrix}, m_{54}=\begin{pmatrix}0&0&0\\1&1&0\\1&0&1\end{pmatrix}, m_{55}=\begin{pmatrix}0&0&0\\1&1&0\\1&1&0\end{pmatrix}, m_{56}=\begin{pmatrix}0&0&0\\1&1&0\\1&1&1\end{pmatrix},$$

$$m_{57}=\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}, m_{58}=\begin{pmatrix}0&0&0\\1&1&1\\0&0&1\end{pmatrix}, m_{59}=\begin{pmatrix}0&0&0\\1&1&1\\0&1&0\end{pmatrix}, m_{60}=\begin{pmatrix}0&0&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{61}=\begin{pmatrix}0&0&0\\1&1&1\\1&0&0\end{pmatrix}, m_{62}=\begin{pmatrix}0&0&0\\1&1&1\\1&0&1\end{pmatrix}, m_{63}=\begin{pmatrix}0&0&0\\1&1&1\\1&1&0\end{pmatrix}, m_{64}=\begin{pmatrix}0&0&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{65}=\begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix}, m_{66}=\begin{pmatrix}0&0&1\\0&0&0\\0&0&1\end{pmatrix}, m_{67}=\begin{pmatrix}0&0&1\\0&0&0\\0&1&0\end{pmatrix}, m_{68}=\begin{pmatrix}0&0&1\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{69}=\begin{pmatrix}0&0&1\\0&0&0\\1&0&0\end{pmatrix}, m_{70}=\begin{pmatrix}0&0&1\\0&0&0\\1&0&1\end{pmatrix}, m_{71}=\begin{pmatrix}0&0&1\\0&0&0\\1&1&0\end{pmatrix}, m_{72}=\begin{pmatrix}0&0&1\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{73}=\begin{pmatrix}0&0&1\\0&0&1\\0&0&0\end{pmatrix}, m_{74}=\begin{pmatrix}0&0&1\\0&0&1\\0&0&1\end{pmatrix}, m_{75}=\begin{pmatrix}0&0&1\\0&0&1\\0&1&0\end{pmatrix}, m_{76}=\begin{pmatrix}0&0&1\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{77}=\begin{pmatrix}0&0&1\\0&0&1\\1&0&0\end{pmatrix}, m_{78}=\begin{pmatrix}0&0&1\\0&0&1\\1&0&1\end{pmatrix}, m_{79}=\begin{pmatrix}0&0&1\\0&0&1\\1&1&0\end{pmatrix}, m_{80}=\begin{pmatrix}0&0&1\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{81}=\begin{pmatrix}0&0&1\\0&1&0\\0&0&0\end{pmatrix}, m_{82}=\begin{pmatrix}0&0&1\\0&1&0\\0&0&1\end{pmatrix}, m_{83}=\begin{pmatrix}0&0&1\\0&1&0\\0&1&0\end{pmatrix}, m_{84}=\begin{pmatrix}0&0&1\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{85}=\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}, m_{86}=\begin{pmatrix}0&0&1\\0&1&0\\1&0&1\end{pmatrix}, m_{87}=\begin{pmatrix}0&0&1\\0&1&0\\1&1&0\end{pmatrix}, m_{88}=\begin{pmatrix}0&0&1\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{89}=\begin{pmatrix}0&0&1\\0&1&1\\0&0&0\end{pmatrix}, m_{90}=\begin{pmatrix}0&0&1\\0&1&1\\0&0&1\end{pmatrix}, m_{91}=\begin{pmatrix}0&0&1\\0&1&1\\0&1&0\end{pmatrix}, m_{92}=\begin{pmatrix}0&0&1\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{93}=\begin{pmatrix}0&0&1\\0&1&1\\1&0&0\end{pmatrix}, m_{94}=\begin{pmatrix}0&0&1\\0&1&1\\1&0&1\end{pmatrix}, m_{95}=\begin{pmatrix}0&0&1\\0&1&1\\1&1&0\end{pmatrix}, m_{96}=\begin{pmatrix}0&0&1\\0&1&1\\1&1&1\end{pmatrix},$$

$m_{97}=\begin{pmatrix}0&0&1\\1&0&0\\0&0&0\end{pmatrix}$, $m_{98}=\begin{pmatrix}0&0&1\\1&0&0\\0&0&1\end{pmatrix}$, $m_{99}=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}$, $m_{100}=\begin{pmatrix}0&0&1\\1&0&0\\0&1&1\end{pmatrix}$,

$m_{101}=\begin{pmatrix}0&0&1\\1&0&0\\1&0&0\end{pmatrix}$, $m_{102}=\begin{pmatrix}0&0&1\\1&0&0\\1&0&1\end{pmatrix}$, $m_{103}=\begin{pmatrix}0&0&1\\1&0&0\\1&1&0\end{pmatrix}$, $m_{104}=\begin{pmatrix}0&0&1\\1&0&0\\1&1&1\end{pmatrix}$,

$m_{105}=\begin{pmatrix}0&0&1\\1&0&1\\0&0&0\end{pmatrix}$, $m_{106}=\begin{pmatrix}0&0&1\\1&0&1\\0&0&1\end{pmatrix}$, $m_{107}=\begin{pmatrix}0&0&1\\1&0&1\\0&1&0\end{pmatrix}$, $m_{108}=\begin{pmatrix}0&0&1\\1&0&1\\0&1&1\end{pmatrix}$,

$m_{109}=\begin{pmatrix}0&0&1\\1&0&1\\1&0&0\end{pmatrix}$, $m_{110}=\begin{pmatrix}0&0&1\\1&0&1\\1&0&1\end{pmatrix}$, $m_{111}=\begin{pmatrix}0&0&1\\1&0&1\\1&1&0\end{pmatrix}$, $m_{112}=\begin{pmatrix}0&0&1\\1&0&1\\1&1&1\end{pmatrix}$,

$m_{113}=\begin{pmatrix}0&0&1\\1&1&0\\0&0&0\end{pmatrix}$, $m_{114}=\begin{pmatrix}0&0&1\\1&1&0\\0&0&1\end{pmatrix}$, $m_{115}=\begin{pmatrix}0&0&1\\1&1&0\\0&1&0\end{pmatrix}$, $m_{116}=\begin{pmatrix}0&0&1\\1&1&0\\0&1&1\end{pmatrix}$,

$m_{117}=\begin{pmatrix}0&0&1\\1&1&0\\1&0&0\end{pmatrix}$, $m_{118}=\begin{pmatrix}0&0&1\\1&1&0\\1&0&1\end{pmatrix}$, $m_{119}=\begin{pmatrix}0&0&1\\1&1&0\\1&1&0\end{pmatrix}$, $m_{120}=\begin{pmatrix}0&0&1\\1&1&0\\1&1&1\end{pmatrix}$,

$m_{121}=\begin{pmatrix}0&0&1\\1&1&1\\0&0&0\end{pmatrix}$, $m_{122}=\begin{pmatrix}0&0&1\\1&1&1\\0&0&1\end{pmatrix}$, $m_{123}=\begin{pmatrix}0&0&1\\1&1&1\\0&1&0\end{pmatrix}$, $m_{124}=\begin{pmatrix}0&0&1\\1&1&1\\0&1&1\end{pmatrix}$,

$m_{125}=\begin{pmatrix}0&0&1\\1&1&1\\1&0&0\end{pmatrix}$, $m_{126}=\begin{pmatrix}0&0&1\\1&1&1\\1&0&1\end{pmatrix}$, $m_{127}=\begin{pmatrix}0&0&1\\1&1&1\\1&1&0\end{pmatrix}$, $m_{128}=\begin{pmatrix}0&0&1\\1&1&1\\1&1&1\end{pmatrix}$,

$m_{129}=\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}$, $m_{130}=\begin{pmatrix}0&1&0\\0&0&0\\0&0&1\end{pmatrix}$, $m_{131}=\begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix}$, $m_{132}=\begin{pmatrix}0&1&0\\0&0&0\\0&1&1\end{pmatrix}$,

$m_{133}=\begin{pmatrix}0&1&0\\0&0&0\\1&0&0\end{pmatrix}$, $m_{134}=\begin{pmatrix}0&1&0\\0&0&0\\1&0&1\end{pmatrix}$, $m_{135}=\begin{pmatrix}0&1&0\\0&0&0\\1&1&0\end{pmatrix}$, $m_{136}=\begin{pmatrix}0&1&0\\0&0&0\\1&1&1\end{pmatrix}$,

$m_{137}=\begin{pmatrix}0&1&0\\0&0&1\\0&0&0\end{pmatrix}$, $m_{138}=\begin{pmatrix}0&1&0\\0&0&1\\0&0&1\end{pmatrix}$, $m_{139}=\begin{pmatrix}0&1&0\\0&0&1\\0&1&0\end{pmatrix}$, $m_{140}=\begin{pmatrix}0&1&0\\0&0&1\\0&1&1\end{pmatrix}$,

$m_{141}=\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}$, $m_{142}=\begin{pmatrix}0&1&0\\0&0&1\\1&0&1\end{pmatrix}$, $m_{143}=\begin{pmatrix}0&1&0\\0&0&1\\1&1&0\end{pmatrix}$, $m_{144}=\begin{pmatrix}0&1&0\\0&0&1\\1&1&1\end{pmatrix}$,

$m_{145}=\begin{pmatrix}0&1&0\\0&1&0\\0&0&0\end{pmatrix}$, $m_{146}=\begin{pmatrix}0&1&0\\0&1&0\\0&0&1\end{pmatrix}$, $m_{147}=\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}$, $m_{148}=\begin{pmatrix}0&1&0\\0&1&0\\0&1&1\end{pmatrix}$,

$$m_{149}=\begin{pmatrix}0&1&0\\0&1&0\\1&0&0\end{pmatrix}, m_{150}=\begin{pmatrix}0&1&0\\0&1&0\\1&0&1\end{pmatrix}, m_{151}=\begin{pmatrix}0&1&0\\0&1&0\\1&1&0\end{pmatrix}, m_{152}=\begin{pmatrix}0&1&0\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{153}=\begin{pmatrix}0&1&0\\0&1&1\\0&0&0\end{pmatrix}, m_{154}=\begin{pmatrix}0&1&0\\0&1&1\\0&0&1\end{pmatrix}, m_{155}=\begin{pmatrix}0&1&0\\0&1&1\\0&1&0\end{pmatrix}, m_{156}=\begin{pmatrix}0&1&0\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{157}=\begin{pmatrix}0&1&0\\0&1&1\\1&0&0\end{pmatrix}, m_{158}=\begin{pmatrix}0&1&0\\0&1&1\\1&0&1\end{pmatrix}, m_{159}=\begin{pmatrix}0&1&0\\0&1&1\\1&1&0\end{pmatrix}, m_{160}=\begin{pmatrix}0&1&0\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{161}=\begin{pmatrix}0&1&0\\1&0&0\\0&0&0\end{pmatrix}, m_{162}=\begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix}, m_{163}=\begin{pmatrix}0&1&0\\1&0&0\\0&1&0\end{pmatrix}, m_{164}=\begin{pmatrix}0&1&0\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{165}=\begin{pmatrix}0&1&0\\1&0&0\\1&0&0\end{pmatrix}, m_{166}=\begin{pmatrix}0&1&0\\1&0&0\\1&0&1\end{pmatrix}, m_{167}=\begin{pmatrix}0&1&0\\1&0&0\\1&1&0\end{pmatrix}, m_{168}=\begin{pmatrix}0&1&0\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{169}=\begin{pmatrix}0&1&0\\1&0&1\\0&0&0\end{pmatrix}, m_{170}=\begin{pmatrix}0&1&0\\1&0&1\\0&0&1\end{pmatrix}, m_{171}=\begin{pmatrix}0&1&0\\1&0&1\\0&1&0\end{pmatrix}, m_{172}=\begin{pmatrix}0&1&0\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{173}=\begin{pmatrix}0&1&0\\1&0&1\\1&0&0\end{pmatrix}, m_{174}=\begin{pmatrix}0&1&0\\1&0&1\\1&0&1\end{pmatrix}, m_{175}=\begin{pmatrix}0&1&0\\1&0&1\\1&1&0\end{pmatrix}, m_{176}=\begin{pmatrix}0&1&0\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{177}=\begin{pmatrix}0&1&0\\1&1&1\\0&0&0\end{pmatrix}, m_{178}=\begin{pmatrix}0&1&0\\1&1&1\\0&0&1\end{pmatrix}, m_{179}=\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}, m_{180}=\begin{pmatrix}0&1&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{181}=\begin{pmatrix}0&1&0\\1&1&1\\1&0&0\end{pmatrix}, m_{182}=\begin{pmatrix}0&1&0\\1&1&1\\1&0&1\end{pmatrix}, m_{183}=\begin{pmatrix}0&1&9\\1&1&1\\1&1&0\end{pmatrix}, m_{184}=\begin{pmatrix}0&1&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{185}=\begin{pmatrix}0&1&0\\1&1&1\\0&0&0\end{pmatrix}, m_{186}=\begin{pmatrix}0&1&0\\1&1&1\\0&0&1\end{pmatrix}, m_{187}=\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}, m_{188}=\begin{pmatrix}0&1&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{189}=\begin{pmatrix}0&1&0\\1&1&1\\1&0&0\end{pmatrix}, m_{190}=\begin{pmatrix}0&1&0\\1&1&1\\1&0&1\end{pmatrix}, m_{191}=\begin{pmatrix}0&1&0\\1&1&1\\1&1&0\end{pmatrix}, m_{192}=\begin{pmatrix}0&1&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{193}=\begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix}, m_{194}=\begin{pmatrix}0&1&1\\0&0&0\\0&0&1\end{pmatrix}, m_{195}=\begin{pmatrix}0&1&1\\0&0&0\\0&1&0\end{pmatrix}, m_{196}=\begin{pmatrix}0&1&1\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{197}=\begin{pmatrix}0&1&1\\0&0&0\\1&0&0\end{pmatrix}, m_{198}=\begin{pmatrix}0&1&1\\0&0&0\\1&0&1\end{pmatrix}, m_{199}=\begin{pmatrix}0&1&1\\0&0&0\\1&1&0\end{pmatrix}, m_{200}=\begin{pmatrix}0&1&1\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{201}=\begin{pmatrix}0&1&1\\0&0&1\\0&0&0\end{pmatrix}, m_{202}=\begin{pmatrix}0&1&1\\0&0&1\\0&0&1\end{pmatrix}, m_{203}=\begin{pmatrix}0&1&1\\0&0&1\\0&1&0\end{pmatrix}, m_{204}=\begin{pmatrix}0&1&1\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{205}=\begin{pmatrix}0&1&1\\0&0&1\\1&0&0\end{pmatrix}, m_{206}=\begin{pmatrix}0&1&1\\0&0&1\\1&0&1\end{pmatrix}, m_{207}=\begin{pmatrix}0&1&1\\0&0&1\\1&1&0\end{pmatrix}, m_{208}=\begin{pmatrix}0&1&1\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{209}=\begin{pmatrix}0&1&1\\0&1&0\\0&0&0\end{pmatrix}, m_{210}=\begin{pmatrix}0&1&1\\0&1&0\\0&0&1\end{pmatrix}, m_{211}=\begin{pmatrix}0&1&1\\0&1&0\\0&1&0\end{pmatrix}, m_{212}=\begin{pmatrix}0&1&1\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{213}=\begin{pmatrix}0&1&1\\0&1&0\\1&0&0\end{pmatrix}, m_{214}=\begin{pmatrix}0&1&1\\0&1&0\\1&0&1\end{pmatrix}, m_{215}=\begin{pmatrix}0&1&1\\0&1&0\\1&1&0\end{pmatrix}, m_{216}=\begin{pmatrix}0&1&1\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{217}=\begin{pmatrix}0&1&1\\0&1&1\\0&0&0\end{pmatrix}, m_{218}=\begin{pmatrix}0&1&1\\0&1&1\\0&0&1\end{pmatrix}, m_{219}=\begin{pmatrix}0&1&1\\0&1&1\\0&1&0\end{pmatrix}, m_{220}=\begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{221}=\begin{pmatrix}0&1&1\\0&1&1\\1&0&0\end{pmatrix}, m_{222}=\begin{pmatrix}0&1&1\\0&1&1\\1&0&1\end{pmatrix}, m_{223}=\begin{pmatrix}0&1&1\\0&1&1\\1&1&0\end{pmatrix}, m_{224}=\begin{pmatrix}0&1&1\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{225}=\begin{pmatrix}0&1&1\\1&0&0\\0&0&0\end{pmatrix}, m_{226}=\begin{pmatrix}0&1&1\\1&0&0\\0&0&1\end{pmatrix}, m_{227}=\begin{pmatrix}0&1&1\\1&0&0\\0&1&0\end{pmatrix}, m_{228}=\begin{pmatrix}0&1&1\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{229}=\begin{pmatrix}0&1&1\\1&0&0\\1&0&0\end{pmatrix}, m_{230}=\begin{pmatrix}0&1&1\\1&0&0\\1&0&1\end{pmatrix}, m_{231}=\begin{pmatrix}0&1&1\\1&0&0\\1&1&0\end{pmatrix}, m_{232}=\begin{pmatrix}0&1&1\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{233}=\begin{pmatrix}0&1&1\\1&0&1\\0&0&0\end{pmatrix}, m_{234}=\begin{pmatrix}0&1&1\\1&0&1\\0&0&1\end{pmatrix}, m_{235}=\begin{pmatrix}0&1&1\\1&0&1\\0&1&0\end{pmatrix}, m_{236}=\begin{pmatrix}0&1&1\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{237}=\begin{pmatrix}0&1&1\\1&0&1\\1&0&0\end{pmatrix}, m_{238}=\begin{pmatrix}0&1&1\\1&0&1\\1&0&1\end{pmatrix}, m_{239}=\begin{pmatrix}0&1&1\\1&0&1\\1&1&0\end{pmatrix}, m_{240}=\begin{pmatrix}0&1&1\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{241}=\begin{pmatrix}0&1&1\\1&1&0\\0&0&0\end{pmatrix}, m_{242}=\begin{pmatrix}0&1&1\\1&1&0\\0&0&1\end{pmatrix}, m_{243}=\begin{pmatrix}0&1&1\\1&1&0\\0&1&0\end{pmatrix}, m_{244}=\begin{pmatrix}0&1&1\\1&1&0\\0&1&1\end{pmatrix},$$

$$m_{245}=\begin{pmatrix}0&1&1\\1&1&0\\1&0&0\end{pmatrix}, m_{246}=\begin{pmatrix}0&1&1\\1&1&0\\1&0&1\end{pmatrix}, m_{247}=\begin{pmatrix}0&1&1\\1&1&0\\1&1&0\end{pmatrix}, m_{248}=\begin{pmatrix}0&1&1\\1&1&0\\1&1&1\end{pmatrix},$$

$$m_{249}=\begin{pmatrix}0&1&1\\1&1&1\\0&0&0\end{pmatrix}, m_{250}=\begin{pmatrix}0&1&1\\1&1&1\\0&0&1\end{pmatrix}, m_{251}=\begin{pmatrix}0&1&1\\1&1&1\\0&1&0\end{pmatrix}, m_{252}=\begin{pmatrix}0&1&1\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{253}=\begin{pmatrix}0&1&1\\1&1&1\\1&0&0\end{pmatrix}, m_{254}=\begin{pmatrix}0&1&1\\1&1&1\\1&0&1\end{pmatrix}, m_{255}=\begin{pmatrix}0&1&1\\1&1&1\\1&1&0\end{pmatrix}, m_{256}=\begin{pmatrix}0&1&1\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{257}=\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}, m_{258}=\begin{pmatrix}1&0&0\\0&0&0\\0&0&1\end{pmatrix}, m_{259}=\begin{pmatrix}1&0&0\\0&0&0\\0&1&0\end{pmatrix}, m_{260}=\begin{pmatrix}1&0&0\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{261}=\begin{pmatrix}1&0&0\\0&0&0\\1&0&0\end{pmatrix}, m_{262}=\begin{pmatrix}1&0&0\\0&0&0\\1&0&1\end{pmatrix}, m_{263}=\begin{pmatrix}1&0&0\\0&0&0\\1&1&0\end{pmatrix}, m_{264}=\begin{pmatrix}1&0&0\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{265}=\begin{pmatrix}1&0&0\\0&0&1\\0&0&0\end{pmatrix}, m_{266}=\begin{pmatrix}1&0&0\\0&0&1\\0&0&1\end{pmatrix}, m_{267}=\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}, m_{268}=\begin{pmatrix}1&0&0\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{269}=\begin{pmatrix}1&0&0\\0&0&1\\1&0&0\end{pmatrix}, m_{270}=\begin{pmatrix}1&0&0\\0&0&1\\1&0&1\end{pmatrix}, m_{271}=\begin{pmatrix}1&0&0\\0&0&1\\1&1&0\end{pmatrix}, m_{272}=\begin{pmatrix}1&0&0\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{273}=\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}, m_{274}=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}, m_{275}=\begin{pmatrix}1&0&0\\0&1&0\\0&1&0\end{pmatrix}, m_{276}=\begin{pmatrix}1&0&0\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{277}=\begin{pmatrix}1&0&0\\0&1&0\\1&0&0\end{pmatrix}, m_{278}=\begin{pmatrix}1&0&0\\0&1&0\\1&0&1\end{pmatrix}, m_{279}=\begin{pmatrix}1&0&0\\0&1&0\\1&1&0\end{pmatrix}, m_{280}=\begin{pmatrix}1&0&0\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{281}=\begin{pmatrix}1&0&0\\0&1&1\\0&0&0\end{pmatrix}, m_{282}=\begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}, m_{283}=\begin{pmatrix}1&0&0\\0&1&1\\0&1&0\end{pmatrix}, m_{284}=\begin{pmatrix}1&0&0\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{285}=\begin{pmatrix}1&0&0\\0&1&1\\1&0&0\end{pmatrix}, m_{286}=\begin{pmatrix}1&0&0\\0&1&1\\1&0&1\end{pmatrix}, m_{287}=\begin{pmatrix}1&0&0\\0&1&1\\1&1&0\end{pmatrix}, m_{288}=\begin{pmatrix}1&0&0\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{289}=\begin{pmatrix}1&0&0\\1&0&0\\0&0&0\end{pmatrix}, m_{290}=\begin{pmatrix}1&0&0\\1&0&0\\0&0&1\end{pmatrix}, m_{291}=\begin{pmatrix}1&0&0\\1&0&0\\0&1&0\end{pmatrix}, m_{292}=\begin{pmatrix}1&0&0\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{293}=\begin{pmatrix}1&0&0\\1&0&0\\1&0&0\end{pmatrix}, m_{294}=\begin{pmatrix}1&0&0\\1&0&0\\1&0&1\end{pmatrix}, m_{295}=\begin{pmatrix}1&0&0\\1&0&0\\1&1&0\end{pmatrix}, m_{296}=\begin{pmatrix}1&0&0\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{297}=\begin{pmatrix}1&0&0\\1&0&1\\0&0&0\end{pmatrix}, m_{298}=\begin{pmatrix}1&0&0\\1&0&1\\0&0&1\end{pmatrix}, m_{299}=\begin{pmatrix}1&0&0\\1&0&1\\0&1&0\end{pmatrix}, m_{300}=\begin{pmatrix}1&0&0\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{301}=\begin{pmatrix}1&0&0\\1&0&1\\1&0&0\end{pmatrix}, m_{302}=\begin{pmatrix}1&0&0\\1&0&1\\1&0&1\end{pmatrix}, m_{303}=\begin{pmatrix}1&0&0\\1&0&1\\1&1&0\end{pmatrix}, m_{304}=\begin{pmatrix}1&0&0\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{305}=\begin{pmatrix}1&0&0\\1&1&0\\0&0&0\end{pmatrix}, m_{306}=\begin{pmatrix}1&0&0\\1&1&0\\0&0&1\end{pmatrix}, m_{307}=\begin{pmatrix}1&0&0\\1&1&0\\0&1&0\end{pmatrix}, m_{308}=\begin{pmatrix}1&0&0\\1&1&0\\0&1&1\end{pmatrix},$$

$$m_{309}=\begin{pmatrix}1&0&0\\1&1&0\\1&0&0\end{pmatrix}, m_{310}=\begin{pmatrix}1&0&0\\1&1&0\\1&0&1\end{pmatrix}, m_{311}=\begin{pmatrix}1&0&0\\1&1&0\\1&1&0\end{pmatrix}, m_{312}=\begin{pmatrix}1&0&0\\1&1&0\\1&1&1\end{pmatrix},$$

$$m_{313}=\begin{pmatrix}1&0&0\\1&1&1\\0&0&0\end{pmatrix}, m_{314}=\begin{pmatrix}1&0&0\\1&1&1\\0&0&1\end{pmatrix}, m_{315}=\begin{pmatrix}1&0&0\\1&1&1\\0&1&0\end{pmatrix}, m_{316}=\begin{pmatrix}1&0&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{317}=\begin{pmatrix}1&0&0\\1&1&1\\1&0&0\end{pmatrix}, m_{318}=\begin{pmatrix}1&0&0\\1&1&1\\1&0&1\end{pmatrix}, m_{319}=\begin{pmatrix}1&0&0\\1&1&1\\1&1&0\end{pmatrix}, m_{320}=\begin{pmatrix}1&0&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{321}=\begin{pmatrix}1&0&1\\0&0&0\\0&0&0\end{pmatrix}, m_{322}=\begin{pmatrix}1&0&1\\0&0&0\\0&0&1\end{pmatrix}, m_{323}=\begin{pmatrix}1&0&1\\0&0&0\\0&1&0\end{pmatrix}, m_{324}=\begin{pmatrix}1&0&1\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{325}=\begin{pmatrix}1&0&1\\0&0&0\\1&0&0\end{pmatrix}, m_{326}=\begin{pmatrix}1&0&1\\0&0&0\\1&0&1\end{pmatrix}, m_{327}=\begin{pmatrix}1&0&1\\0&0&0\\1&1&0\end{pmatrix}, m_{328}=\begin{pmatrix}1&0&1\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{329}=\begin{pmatrix}1&0&1\\0&0&1\\0&0&0\end{pmatrix}, m_{330}=\begin{pmatrix}1&0&1\\0&0&1\\0&0&1\end{pmatrix}, m_{331}=\begin{pmatrix}1&0&1\\0&0&1\\0&1&0\end{pmatrix}, m_{332}=\begin{pmatrix}1&0&1\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{333}=\begin{pmatrix}1&0&1\\0&0&1\\1&0&0\end{pmatrix}, m_{334}=\begin{pmatrix}1&0&1\\0&0&1\\1&0&1\end{pmatrix}, m_{335}=\begin{pmatrix}1&0&1\\0&0&1\\1&1&0\end{pmatrix}, m_{336}=\begin{pmatrix}1&0&1\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{337}=\begin{pmatrix}1&0&1\\0&1&0\\0&0&0\end{pmatrix}, m_{338}=\begin{pmatrix}1&0&1\\0&1&0\\0&0&1\end{pmatrix}, m_{339}=\begin{pmatrix}1&0&1\\0&1&0\\0&1&0\end{pmatrix}, m_{340}=\begin{pmatrix}1&0&1\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{341}=\begin{pmatrix}1&0&1\\0&1&0\\1&0&0\end{pmatrix}, m_{342}=\begin{pmatrix}1&0&1\\0&1&0\\1&0&1\end{pmatrix}, m_{343}=\begin{pmatrix}1&0&1\\0&1&0\\1&1&0\end{pmatrix}, m_{344}=\begin{pmatrix}1&0&1\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{345}=\begin{pmatrix}1&0&1\\0&1&1\\0&0&0\end{pmatrix}, m_{346}=\begin{pmatrix}1&0&1\\0&1&1\\0&0&1\end{pmatrix}, m_{347}=\begin{pmatrix}1&0&1\\0&1&1\\0&1&0\end{pmatrix}, m_{348}=\begin{pmatrix}1&0&1\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{349}=\begin{pmatrix}1&0&1\\0&1&1\\1&0&0\end{pmatrix}, m_{350}=\begin{pmatrix}1&0&1\\0&1&1\\1&0&1\end{pmatrix}, m_{351}=\begin{pmatrix}1&0&1\\0&1&1\\1&1&0\end{pmatrix}, m_{352}=\begin{pmatrix}1&0&1\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{353}=\begin{pmatrix}1&0&1\\1&0&0\\0&0&0\end{pmatrix}, m_{354}=\begin{pmatrix}1&0&1\\1&0&0\\0&0&1\end{pmatrix}, m_{355}=\begin{pmatrix}1&0&1\\1&0&0\\0&1&0\end{pmatrix}, m_{356}=\begin{pmatrix}1&0&1\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{357}=\begin{pmatrix}1&0&1\\1&0&0\\1&0&0\end{pmatrix}, m_{358}=\begin{pmatrix}1&0&1\\1&0&0\\1&0&1\end{pmatrix}, m_{359}=\begin{pmatrix}1&0&1\\1&0&0\\1&1&0\end{pmatrix}, m_{360}=\begin{pmatrix}1&0&1\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{361}=\begin{pmatrix}1&0&1\\1&0&1\\0&0&0\end{pmatrix}, m_{362}=\begin{pmatrix}1&0&1\\1&0&1\\0&0&1\end{pmatrix}, m_{363}=\begin{pmatrix}1&0&1\\1&0&1\\0&1&0\end{pmatrix}, m_{364}=\begin{pmatrix}1&0&1\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{365}=\begin{pmatrix}1&0&1\\1&0&1\\1&0&0\end{pmatrix}, m_{366}=\begin{pmatrix}1&0&1\\1&0&1\\1&0&1\end{pmatrix}, m_{367}=\begin{pmatrix}1&0&1\\1&0&1\\1&1&0\end{pmatrix}, m_{368}=\begin{pmatrix}1&0&1\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{369}=\begin{pmatrix}1&0&1\\1&1&0\\0&0&0\end{pmatrix}, m_{370}=\begin{pmatrix}1&0&1\\1&1&0\\0&0&1\end{pmatrix}, m_{371}=\begin{pmatrix}1&0&1\\1&1&0\\0&1&0\end{pmatrix}, m_{372}=\begin{pmatrix}1&0&1\\1&1&0\\0&1&1\end{pmatrix},$$

$$m_{373}=\begin{pmatrix}1&0&1\\1&1&0\\1&0&0\end{pmatrix}, m_{374}=\begin{pmatrix}1&0&1\\1&1&0\\1&0&1\end{pmatrix}, m_{375}=\begin{pmatrix}1&0&1\\1&1&0\\1&1&0\end{pmatrix}, m_{376}=\begin{pmatrix}1&0&1\\1&1&0\\1&1&1\end{pmatrix},$$

$$m_{377}=\begin{pmatrix}1&0&1\\1&1&1\\0&0&0\end{pmatrix}, m_{378}=\begin{pmatrix}1&0&1\\1&1&1\\0&0&1\end{pmatrix}, m_{379}=\begin{pmatrix}1&0&1\\1&1&1\\0&1&0\end{pmatrix}, m_{380}=\begin{pmatrix}1&0&1\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{381}=\begin{pmatrix}1&0&1\\1&1&1\\1&0&0\end{pmatrix}, m_{382}=\begin{pmatrix}1&0&1\\1&1&1\\1&0&1\end{pmatrix}, m_{383}=\begin{pmatrix}1&0&1\\1&1&1\\1&1&0\end{pmatrix}, m_{384}=\begin{pmatrix}1&0&1\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{385}=\begin{pmatrix}1&1&0\\0&0&0\\0&0&0\end{pmatrix}, m_{386}=\begin{pmatrix}1&1&0\\0&0&0\\0&0&1\end{pmatrix}, m_{387}=\begin{pmatrix}1&1&0\\0&0&0\\0&1&0\end{pmatrix}, m_{388}=\begin{pmatrix}1&1&0\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{389}=\begin{pmatrix}1&1&0\\0&0&0\\1&0&0\end{pmatrix}, m_{390}=\begin{pmatrix}1&1&0\\0&0&0\\1&0&1\end{pmatrix}, m_{391}=\begin{pmatrix}1&1&0\\0&0&0\\1&1&0\end{pmatrix}, m_{392}=\begin{pmatrix}1&1&0\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{393}=\begin{pmatrix}1&1&0\\0&0&1\\0&0&0\end{pmatrix}, m_{394}=\begin{pmatrix}1&1&0\\0&0&1\\0&0&1\end{pmatrix}, m_{395}=\begin{pmatrix}1&1&0\\0&0&1\\0&1&0\end{pmatrix}, m_{396}=\begin{pmatrix}1&1&0\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{397}=\begin{pmatrix}1&1&0\\0&0&1\\1&0&0\end{pmatrix}, m_{398}=\begin{pmatrix}1&1&0\\0&0&1\\1&0&1\end{pmatrix}, m_{399}=\begin{pmatrix}1&1&0\\0&0&1\\1&1&0\end{pmatrix}, m_{400}=\begin{pmatrix}1&1&0\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{401}=\begin{pmatrix}1&1&0\\0&1&0\\0&0&0\end{pmatrix}, m_{402}=\begin{pmatrix}1&1&0\\0&1&0\\0&0&1\end{pmatrix}, m_{403}=\begin{pmatrix}1&1&0\\0&1&0\\0&1&0\end{pmatrix}, m_{404}=\begin{pmatrix}1&1&0\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{405}=\begin{pmatrix}1&1&0\\0&1&0\\1&0&0\end{pmatrix}, m_{406}=\begin{pmatrix}1&1&0\\0&1&0\\1&0&1\end{pmatrix}, m_{407}=\begin{pmatrix}1&1&0\\0&1&0\\1&1&0\end{pmatrix}, m_{408}=\begin{pmatrix}1&1&0\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{409}=\begin{pmatrix}1&1&0\\0&1&1\\0&0&0\end{pmatrix}, m_{410}=\begin{pmatrix}1&1&0\\0&1&1\\0&0&1\end{pmatrix}, m_{411}=\begin{pmatrix}1&1&0\\0&1&1\\0&1&0\end{pmatrix}, m_{412}=\begin{pmatrix}1&1&0\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{413}=\begin{pmatrix}1&1&0\\0&1&1\\1&0&0\end{pmatrix}, m_{414}=\begin{pmatrix}1&1&0\\0&1&1\\1&0&1\end{pmatrix}, m_{415}=\begin{pmatrix}1&1&0\\0&1&1\\1&1&0\end{pmatrix}, m_{416}=\begin{pmatrix}1&1&0\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{417}=\begin{pmatrix}1&1&0\\1&0&0\\0&0&0\end{pmatrix}, m_{418}=\begin{pmatrix}1&1&0\\1&0&0\\0&0&1\end{pmatrix}, m_{419}=\begin{pmatrix}1&1&0\\1&0&0\\0&1&0\end{pmatrix}, m_{420}=\begin{pmatrix}1&1&0\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{421}=\begin{pmatrix}1&1&0\\1&0&0\\1&0&0\end{pmatrix}, m_{422}=\begin{pmatrix}1&1&0\\1&0&0\\1&0&1\end{pmatrix}, m_{423}=\begin{pmatrix}1&1&0\\1&0&0\\1&1&0\end{pmatrix}, m_{424}=\begin{pmatrix}1&1&0\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{425}=\begin{pmatrix}1&1&0\\1&0&1\\0&0&0\end{pmatrix}, m_{426}=\begin{pmatrix}1&1&0\\1&0&1\\0&0&1\end{pmatrix}, m_{427}=\begin{pmatrix}1&1&0\\1&0&1\\0&1&0\end{pmatrix}, m_{428}=\begin{pmatrix}1&1&0\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{429}=\begin{pmatrix}1&1&0\\1&0&1\\1&0&0\end{pmatrix}, m_{430}=\begin{pmatrix}1&1&0\\1&0&1\\1&0&1\end{pmatrix}, m_{431}=\begin{pmatrix}1&1&0\\1&0&1\\1&1&0\end{pmatrix}, m_{432}=\begin{pmatrix}1&1&0\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{433}=\begin{pmatrix}1&1&0\\1&1&1\\0&0&0\end{pmatrix}, m_{434}=\begin{pmatrix}1&1&0\\1&1&1\\0&0&1\end{pmatrix}, m_{435}=\begin{pmatrix}1&1&0\\1&1&1\\0&1&0\end{pmatrix}, m_{436}=\begin{pmatrix}1&1&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{437}=\begin{pmatrix}1&1&0\\1&1&1\\1&0&0\end{pmatrix}, m_{438}=\begin{pmatrix}1&1&0\\1&1&1\\1&0&1\end{pmatrix}, m_{439}=\begin{pmatrix}1&1&9\\1&1&1\\1&1&0\end{pmatrix}, m_{440}=\begin{pmatrix}1&1&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{441}=\begin{pmatrix}1&1&0\\1&1&1\\0&0&0\end{pmatrix}, m_{442}=\begin{pmatrix}1&1&0\\1&1&1\\0&0&1\end{pmatrix}, m_{443}=\begin{pmatrix}1&1&0\\1&1&1\\0&1&0\end{pmatrix}, m_{444}=\begin{pmatrix}1&1&0\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{445}=\begin{pmatrix}1&1&0\\1&1&1\\1&0&0\end{pmatrix}, m_{446}=\begin{pmatrix}1&1&0\\1&1&1\\1&0&1\end{pmatrix}, m_{447}=\begin{pmatrix}1&1&0\\1&1&1\\1&1&0\end{pmatrix}, m_{448}=\begin{pmatrix}1&1&0\\1&1&1\\1&1&1\end{pmatrix},$$

$$m_{449}=\begin{pmatrix}1&1&1\\0&0&0\\0&0&0\end{pmatrix}, m_{450}=\begin{pmatrix}1&1&1\\0&0&0\\0&0&1\end{pmatrix}, m_{451}=\begin{pmatrix}1&1&1\\0&0&0\\0&1&0\end{pmatrix}, m_{452}=\begin{pmatrix}1&1&1\\0&0&0\\0&1&1\end{pmatrix},$$

$$m_{453}=\begin{pmatrix}1&1&1\\0&0&0\\1&0&0\end{pmatrix}, m_{454}=\begin{pmatrix}1&1&1\\0&0&0\\1&0&1\end{pmatrix}, m_{455}=\begin{pmatrix}1&1&1\\0&0&0\\1&1&0\end{pmatrix}, m_{456}=\begin{pmatrix}1&1&1\\0&0&0\\1&1&1\end{pmatrix},$$

$$m_{457}=\begin{pmatrix}1&1&1\\0&0&1\\0&0&0\end{pmatrix}, m_{458}=\begin{pmatrix}1&1&1\\0&0&1\\0&0&1\end{pmatrix}, m_{459}=\begin{pmatrix}1&1&1\\0&0&1\\0&1&0\end{pmatrix}, m_{460}=\begin{pmatrix}1&1&1\\0&0&1\\0&1&1\end{pmatrix},$$

$$m_{461}=\begin{pmatrix}1&1&1\\0&0&1\\1&0&0\end{pmatrix}, m_{462}=\begin{pmatrix}1&1&1\\0&0&1\\1&0&1\end{pmatrix}, m_{463}=\begin{pmatrix}1&1&1\\0&0&1\\1&1&0\end{pmatrix}, m_{464}=\begin{pmatrix}1&1&1\\0&0&1\\1&1&1\end{pmatrix},$$

$$m_{465}=\begin{pmatrix}1&1&1\\0&1&0\\0&0&0\end{pmatrix}, m_{466}=\begin{pmatrix}1&1&1\\0&1&0\\0&0&1\end{pmatrix}, m_{467}=\begin{pmatrix}1&1&1\\0&1&0\\0&1&0\end{pmatrix}, m_{468}=\begin{pmatrix}1&1&1\\0&1&0\\0&1&1\end{pmatrix},$$

$$m_{469}=\begin{pmatrix}1&1&1\\0&1&0\\1&0&0\end{pmatrix}, m_{470}=\begin{pmatrix}1&1&1\\0&1&0\\1&0&1\end{pmatrix}, m_{471}=\begin{pmatrix}1&1&1\\0&1&0\\1&1&0\end{pmatrix}, m_{472}=\begin{pmatrix}1&1&1\\0&1&0\\1&1&1\end{pmatrix},$$

$$m_{473}=\begin{pmatrix}1&1&1\\0&1&1\\0&0&0\end{pmatrix}, m_{474}=\begin{pmatrix}1&1&1\\0&1&1\\0&0&1\end{pmatrix}, m_{475}=\begin{pmatrix}1&1&1\\0&1&1\\0&1&0\end{pmatrix}, m_{476}=\begin{pmatrix}1&1&1\\0&1&1\\0&1&1\end{pmatrix},$$

$$m_{477}=\begin{pmatrix}1&1&1\\0&1&1\\1&0&0\end{pmatrix}, m_{478}=\begin{pmatrix}1&1&1\\0&1&1\\1&0&1\end{pmatrix}, m_{479}=\begin{pmatrix}1&1&1\\0&1&1\\1&1&0\end{pmatrix}, m_{480}=\begin{pmatrix}1&1&1\\0&1&1\\1&1&1\end{pmatrix},$$

$$m_{481}=\begin{pmatrix}1&1&1\\1&0&0\\0&0&0\end{pmatrix}, m_{482}=\begin{pmatrix}1&1&1\\1&0&0\\0&0&1\end{pmatrix}, m_{483}=\begin{pmatrix}1&1&1\\1&0&0\\0&1&0\end{pmatrix}, m_{484}=\begin{pmatrix}1&1&1\\1&0&0\\0&1&1\end{pmatrix},$$

$$m_{485}=\begin{pmatrix}1&1&1\\1&0&0\\1&0&0\end{pmatrix}, m_{486}=\begin{pmatrix}1&1&1\\1&0&0\\1&0&1\end{pmatrix}, m_{487}=\begin{pmatrix}1&1&1\\1&0&0\\1&1&0\end{pmatrix}, m_{488}=\begin{pmatrix}1&1&1\\1&0&0\\1&1&1\end{pmatrix},$$

$$m_{489}=\begin{pmatrix}1&1&1\\1&0&1\\0&0&0\end{pmatrix}, m_{490}=\begin{pmatrix}1&1&1\\1&0&1\\0&0&1\end{pmatrix}, m_{491}=\begin{pmatrix}1&1&1\\1&0&1\\0&1&0\end{pmatrix}, m_{492}=\begin{pmatrix}1&1&1\\1&0&1\\0&1&1\end{pmatrix},$$

$$m_{493}=\begin{pmatrix}1&1&1\\1&0&1\\1&0&0\end{pmatrix}, m_{494}=\begin{pmatrix}1&1&1\\1&0&1\\1&0&1\end{pmatrix}, m_{495}=\begin{pmatrix}1&1&1\\1&0&1\\1&1&0\end{pmatrix}, m_{496}=\begin{pmatrix}1&1&1\\1&0&1\\1&1&1\end{pmatrix},$$

$$m_{497}=\begin{pmatrix}1&1&1\\1&1&0\\0&0&0\end{pmatrix}, m_{498}=\begin{pmatrix}1&1&1\\1&1&0\\0&0&1\end{pmatrix}, m_{499}=\begin{pmatrix}1&1&1\\1&1&0\\0&1&0\end{pmatrix}, m_{500}=\begin{pmatrix}1&1&1\\1&1&0\\0&1&1\end{pmatrix},$$

$$m_{501}=\begin{pmatrix}1&1&1\\1&1&0\\1&0&0\end{pmatrix}, m_{502}=\begin{pmatrix}1&1&1\\1&1&0\\1&0&1\end{pmatrix}, m_{503}=\begin{pmatrix}1&1&1\\1&1&0\\1&1&0\end{pmatrix}, m_{504}=\begin{pmatrix}1&1&1\\1&1&0\\1&1&1\end{pmatrix},$$

$$m_{505}=\begin{pmatrix}1&1&1\\1&1&1\\0&0&0\end{pmatrix}, m_{506}=\begin{pmatrix}1&1&1\\1&1&1\\0&0&1\end{pmatrix}, m_{507}=\begin{pmatrix}1&1&1\\1&1&1\\0&1&0\end{pmatrix}, m_{508}=\begin{pmatrix}1&1&1\\1&1&1\\0&1&1\end{pmatrix},$$

$$m_{509}=\begin{pmatrix}1&1&1\\1&1&1\\1&0&0\end{pmatrix}, m_{510}=\begin{pmatrix}1&1&1\\1&1&1\\1&0&1\end{pmatrix}, m_{511}=\begin{pmatrix}1&1&1\\1&1&1\\1&1&0\end{pmatrix}, m_{512}=\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}.$$